H. A. Eiselt
C.-L. Sandblom

Operations Research

A Model-Based Approach

Springer

VISIT…

Operations Research

H.A. Eiselt • C.-L. Sandblom

Operations Research

A Model-Based Approach

Springer

Professor Dr. H. A. Eiselt
University of New Brunswick
Faculty of Business Administration
7 Mac Aulay Drive
Fredericton, NB E3B 5A3
Canada
haeiselt@unb.ca

Professor Dr. C.-L. Sandblom
Dalhousie University
Faculty of Industrial Engineering
Halifax, NS B3J 2X4
Canada
carl-louis.sandblom@dal.ca

ISBN 978-3-642-10325-4 e-ISBN 978-3-642-10326-1
DOI 10.1007/978-3-642-10326-1
Springer Heidelberg Dordrecht London New York

Library of Congress Control Number: 2010925338

© Springer-Verlag Berlin Heidelberg 2010
This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilm or in any other way, and storage in data banks. Duplication of this publication or parts thereof is permitted only under the provisions of the German Copyright Law of September 9, 1965, in its current version, and permission for use must always be obtained from Springer. Violations are liable to prosecution under the German Copyright Law.
The use of general descriptive names, registered names, trademarks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

Cover design: WMXDesign GmbH, Heidelberg, Germany

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

“And there is so much more value in learning why a set of conditions exists than simply accepting those conditions and committing them to memory.”

Ben Stein

PREFACE

Since the 1960s, operations research (or, alternatively, management science) has become an indispensable tool in scientific management. In simple words, its goal on the strategic and tactical levels is to aid in decision making and, on the operational level, automate decision making. Its tools are algorithms, procedures that create and improve solutions to a point at which optimal or, at least, satisfactory solutions have been found.

While many texts on the subject emphasize methods, the special focus of this book is on the applications of operations research in practice. Typically, a topic is introduced by means of a description of its applications, a model is formulated and its solution is presented. Then the solution is discussed and its implications for decision making are outlined. We have attempted to maximize the understanding of the topics by using intuitive reasoning while keeping mathematical notation and the description of techniques to a minimum. The exercises are designed to fully explore the material covered in the chapters, without resorting to mind-numbing repetitions and trivialization.

The book is designed for (typically second year) students of business management and industrial engineering. With the appropriate deletions, the material can be used for a one-semester course in the subject, while the complete material will be sufficient for a full-year course. The reasoning and explanations are intuitive throughout. Each algorithm is followed by a numerical example that shows in detail how the method progresses. After presenting the applications and the techniques, each chapter ends with a number of fully solved examples that review the concepts covered in the chapter. Some more technical material has been taken out and is available at the publisher's website.

It is our pleasure to thank all the people who have made this volume possible. Special thanks are due to Dr. Müller of Springer Publishers, who first suggested this volume at one of our meetings. It has been a very pleasurable experience to work with Dr. Müller during the last quarter century on various projects. Thanks are also due to Mrs. Milewski for her technical advice and timely replies to our queries. We would also like to express our gratitude to our assistants #21 (a.k.a.

Courtney Palmer), as well as Jun Zhou and Eric Giacomin for their help in producing the figures, and to the Buddha Man for his meticulous typing. Without the help of all of these individuals, this book would not have seen the light of day. We like to thank all of them.

H.A. Eiselt
C.-L. Sandblom

CONTENTS

Preface vii

1. Introduction to Operations Research 1
1.1 The Nature and History of Operations Research 1
1.2 The Main Elements of Operations Research 4
1.3 The Modeling Process 9

2. Linear Programming 13
2.1 Introduction to Linear Programming 13
2.2 Applications of Linear Programming 18
2.2.1 Production Planning 18
2.2.2 Diet Problems 20
2.2.3 Allocation Problems 28
2.2.4 Employee Scheduling 32
2.2.5 Dynamic Production–Inventory Models 35
2.2.6 Blending Problems 39
2.2.7 Transportation and Assignment Problems 43
Exercises 50
2.3 Graphical Representation and Solution 60
2.3.1 The Graphical Solution Method 60
2.3.2 Special Cases of Linear Programming Problems 70
Exercises 76
2.4 Postoptimality Analyses 78
2.4.1 Graphical Sensitivity Analyses 78
2.4.2 Economic Analysis of an Optimal Solution 92
Exercises 100
2.5 Duality 105
Exercises 112

3. Multiobjective Programming 115
3.1 Vector Optimization 116
3.2 Solution Approaches to Vector Optimization Problems 121
3.3 Goal Programming 124
Exercises 129

4. Integer Programming 135
4.1 Definitions and Basic Concepts 135
4.2 Applications of Integer Programming 140
4.2.1 Cutting Stock Problems 142
4.2.2 Diet Problems Revisited 146
4.2.3 Land Use 148
4.2.4 Modeling Fixed Charges 150
4.2.5 Workload Balancing 152
4.3 Solution Methods for Integer Programming Problems 154
4.3.1 Cutting Plane Methods 154
4.3.2 Branch-and-Bound Methods 155
4.3.3 Heuristic Methods 162
Exercises 165

5. Network Models 177
5.1 Definitions and Conventions 177
5.2 Network Flow Problems 179
5.3 Shortest Path Problems 189
5.4 Spanning Tree Problems 198
5.5 Routing Problems 200
Exercises 205

6. Location Models 217
6.1 The Major Elements of Location Problems 217
6.2 Covering Problems 220
6.2.1 The Location Set Covering Problem 221
6.2.2 The Maximal Covering Location Problem 227
6.3 Center Problems 230
6.3.1 1-Center Problems 231
6.3.2 *p*-Center Problems 233
6.4 Median Problems 235
6.4.1 Minisum Problems in the Plane 235
6.4.2 Minisum Problems in Networks 240
6.5 Other Location Problems 244
Exercises 247

7. Project Networks 257
7.1 The Critical Path Method 258
7.2 Project Acceleration 266

7.3 Project Planning with Resources 272
7.4 The PERT Method 275
Exercises 280

8. Machine Scheduling 287
8.1 Basic Concepts of Machine Scheduling 288
8.2 Single Machine Scheduling 290
8.3 Parallel Machine Scheduling 294
8.4 Dedicated Machine Scheduling 297
Exercises 301

9. Decision Analysis 305
9.1 Introduction to Decision Analysis 305
9.2 Visualizations of Decision Problems 307
9.3 Decision Rules Under Uncertainty and Risk 310
9.4 Sensitivity Analyses 316
9.5 Decision Trees and the Value of Information 319
9.6 Utility Theory 327
Exercises 328

10. Inventory Models 339
10.1 Basic Concepts in Inventory Planning 339
10.2 The Economic Order Quantity (*EOQ*) Model 343
10.3 The Economic Order Quantity with Positive Lead Time 346
10.4 The Economic Order Quantity with Backorders 349
10.5 The Economic Order Quantity with Quantity Discounts 352
10.6 The Production Lot Size Model 355
10.7 The Economic Order Quantity with Stochastic Lead Time Demand 357
10.7.1 A Model that Optimizes the Reorder Point 359
10.7.2 A Stochastic Model with Simultaneous Computation of Order Quantity and Reorder Point 360
10.8 Extensions of the Basic Inventory Models 362
Exercises 363

11. Stochastic Processes and Markov Chains 367
11.1 Basic Ideas and Concepts 367
11.2 Steady-State Solutions 372
11.3 Decision Making with Markov Chains 373
Exercises 376

12. Waiting Line Models 379
12.1 Basic Queuing Models 380
12.2 Optimization in Queuing 388
Exercises 392

13. Simulation 395
13.1 Introduction to Simulation 395
13.2 Random Numbers and their Generation 397
13.3 Examples of Simulations 402
13.3.1 Simulation of a Waiting Line System 402
13.3.2 Simulation of an Inventory System 405
Exercises 410

Appendices

A. Heuristic Algorithms 417

B. Vectors and Matrices 427

C. Systems of Simultaneous Linear Equations 429

D. Probability and Statistics 433

References 441

Subject Index 443

1 Introduction to Operations Research

In its first section, this introductory chapter first introduces operations research as a discipline. It defines its function and then traces its roots to its beginnings. The second section highlights some of the main elements of operations research and discusses a number of potential difficulties and pitfalls. Finally, the third section of this chapter suggests an eight-step procedure for the modeling process.

1.1 The Nature and History of Operations Research

The subject matter, *operations research* or *management science* (even though there may be philosophical differences, we use the two terms interchangeably), has been defined by many researchers in the field. Definitions range from "a scientific approach to decision making," to "the use of quantitative tools for systems that originate from real life," "scientific decision making," and others. In the mid-1970s, the Operations Research Society of America (then one of the two large professional societies in the field) defined the subject matter as follows:

> "Operations Research is concerned with scientifically deciding how to best design and operate man-machine systems usually under conditions requiring the allocation of scarce resources."

Today, the Institute for Operations Research and Management Science (INFORMS) markets operations research as the "science of better." What all of this essentially means is that the science uses indeed quantitative techniques to make and prepare decisions, by determining the most efficient way to act under given circumstances. In other words, rather than throwing large amounts of resources (such as money) at a problem, operations research will determine ways to do things more efficiently.

Rather than being restricted to being a toolkit for quantitative planners, operations research is much more: it is a way of thinking that does not just "do things," but, during each step of the way, attempts to do them more efficiently: the waitress, who

H.A. Eiselt and C.-L. Sandblom, *Operations Research: A Model-Based Approach*,
DOI 10.1007/978-3-642-10326-1_1, © Springer-Verlag Berlin Heidelberg 2010

provides coffee refills along the way rather than making special trips; the personnel manager who (re-) assigns employees so as to either minimize the number of employees needed, or to schedule employees to shifts, so as to make them more pleasant; the municipal planner, who incorporates the typically widely diverging goals and objectives of multiple constituents or stakeholders when locating a new sewage treatment plants; the project manager, who has to coordinate many different and independent activities. All of these individuals can benefit from the large variety of tools available.

There are a number of obstacles that stand in the way of the extensive use of operations research in practice. One of those obstacles is awareness. If managers were to be able to realize that a problem may possibly benefit from the use of a quantitative analysis, it does not matter at all, whether or not he can perform the analyses himself: there are plenty of specialists out there to do the job. The first step, though, requires someone to simply realize that operations research could be applied to a problem to a problem that presently requires a solution. This is the reason why we have written this book with a strong focus on applications.

When trying to find out where you are and where you are going, it is always a good idea to determine where you come from. The next few paragraphs will highlight some of the main milestones to operations research. Clearly, space limitations require us to cut many corners. We would like to refer to the eminently readable history of operations research by Gass and Assad (2005).

What are usually considered to be early contributions, are usually advances in mathematics or statistics: Diophantine's discourses on integer solutions to linear equations in the third century AD are related to integer programming, Euler's work on the Königsberg bridge problem in 1736 is the first occurrence of graph theory; Pascal, Bernoulli, and Bayes have made major advances in statistics. All results found by these and many other scientists have put down a mathematical and statistical foundation, on which operations research (and many other disciplines) can rest comfortably.

While many authors credit the advances in the military in World War II to the birth of operations research, we believe that the groundwork was laid considerably earlier. F.W. Taylor is often called the "father of scientific management," when he performed his time studies in 1881. His main question was "what is the best way to do a job," which could very well be the motto of operations research. Henry L. Gantt introduced bar charts, "Gantt charts" in today's parlance, for scheduling problems, and Agner Krarup Erlang introduced the discipline of queuing in 1909 when working at the Copenhagen Telephone exchange. The final contribution in the early days was made by F.W. Harris in 1913, when he developed the "economic order quantity" for inventory management, a result that is so robust that it is, in one way or another, used to this day. All of these individuals would today be

referred to as industrial engineers, as their main concern was the smooth functioning of industrial processes.

It is hardly surprising that these early contributions occurred at a time that saw more complex industrial processes (the assembly line is but one example), a tremendous increase in the division of labor, and with it the need for coordination of activities.

Later notable work was performed by John von Neumann in the 1920s, when he introduced the theory of games to the world. Leontief's input – output models and Kantorovich's mathematical planning models for the Soviet economy were main contributions in the 1930s. The 1940s saw Hitchcock's transportation problem, Stigler's diet planning, and the aforementioned advances based on military applications.

However, the main event occurred in August of 1947 when George Bernard Dantzig developed what is now called the simplex method for linear programming. Arguably, no other event has influenced the science of operations research more than this development. Other main developments are due to John F. Nash, who extended von Neumann's results in game theory and proved some main theorems, Bellman's dynamic programming principle in 1950, and Kuhn and Tucker's optimality conditions for nonlinear optimization problems in 1951 (which was later discovered to be a reinvention of work by Karush in 1939). The year 1951 saw not only the first full publication of Dantzig's simplex method in the open literature, but also the first computer-based simplex method. As a matter of fact, the advances in computing hardware and software had a tremendous impact on the advances of operations research. Without the progress made in computer sciences, operations research would not have been able to gain the status it has today.

New results keep pouring in. Starting in 1950, the *Operational Research Quarterly* (later renamed the *Journal of the Operational Research Society*) was the first journal in the field published in the United Kingdom. The first American journal followed in 1952 with the *Journal of the Operations Research Society of America*. Today, many countries have their own operations research journals. To name a few, there are the *European Journal of Operational Research* (the largest operations research journal by size, about 8,000 pages per year), INFOR (Canada), the *OR Spectrum* (Germany), *TOP* (Spain), the *Central European Journal of Operations Research* (Austria), the *Yugoslav Journal of Operations Research*, *Opsearch* (India), *Pesquisa Operacional* (Brazil), and many more. In addition, there are many specialist journals, such as *Computers & Operations Research*, *Mathematical Programming*, *Management Science*, *Naval Research Logistics*, and many others.

And wherever there is a national journal, more often than not there is a national society behind it. Each of these societies have an annual meeting, where researchers

present their latest findings. Again, in addition to these national conferences, there are meetings devoted to special topics such as optimization, logistics, supply chains, location, transportation, scheduling, and many more. It should be apparent by now that the number of contributions can only be described as vast.

1.2 The Main Elements of Operations Research

This section will briefly explain the main elements of operations research. Essentially, operations research is concerned with quantitative models and their solution. This is actually, where some people make the distinction: they claim that while management science is mostly concerned with models, operations research deals mostly with solution methods. The model that we build for a given scenario is a (hopefully close) picture of reality. A model will never include all components a real situation does: features such as some of the choice criteria used by consumers are not fully observable and will have to be ignored, decision makers' risk aversion, while it may be included to some degree in a decision making model, is not completely explainable and will have to be estimated or ignored. It is very important to distinguish between the original real-life problem, and the mathematical model that is built. The step from the problem to the model entails gains and losses: while we are losing some information, we gain solvability and gain insight into the structure of the problem. As one of the founders of the science once wrote "the purpose of mathematical programming [a subtopic of operations research, eds.] is insight, not numbers."

The next issue to be decided upon is the level of aggregation. In conjunction with the decision maker the modeler will have to decide what to include in the model and what to leave out. Comprehensive models are nice and avoid problems with suboptimal solutions, but they are large, labor-intensive, and thus expensive. It depends on the specific situation what level is appropriate. The aforementioned suboptimal solutions may occur, if we were to optimize for only one department of a firm. For instance, the manager of a shipping department or a large firm that manufactured construction equipment consistently hired employees through a temp agency, even though this was much more expensive than hiring people directly. However, temporary employees were paid by headquarters, while the pay for regular employees came directly out of the manager's budget. Thus, hiring temporary employees was optimal for the manager, even though it cost the firm much more money.

The usual way to describe situations for operations research models is to first list everything we *want* to do, and everything that we *have to* do and respect. What we want to achieve (high profit, high levels of customer satisfaction, a large market share, low production costs, or similar) is summarized in our *objective*(*s*). On the other hand, all requirements (such as budget limitations, limitations of a firm's

capabilities with respect to manpower, knowledge base, available machinery, and others) are summarized in the *constraints*. While in most operations research tools, objectives and constraints are clearly separated, this is not always true in reality. Consider, for instance, a simple budget constraint. It will state that the amount of money we can spend in, say, a month cannot exceed the amount of money that we have. We can write this as a formal constraint, as long as we are aware of the fact that a constraint in operations research is absolute. This means that whatever the solution technique is, it will not consider any solution that violates this constraint. In other words, it will consider the constraint as hard. In practice, however, the constraint may not be that hard. We could, for instance, take out a loan and temporarily spend more than we have. While this is necessarily a temporary measure and we would like to avoid it, it is possible in practice. Written as a constraint, however, it is not possible. There are techniques that address this problem by formulating the models differently, allowing short-term deficits, but trying to minimize this. Notice that all of a sudden what we used to clearly consider as a constraint has become part of the objective function.

Another important issue to deal with is the issue of measurement. While many of the features of a model are easily measurable (take, for instance, profits, travel time, processing times, the length and width of a stretch of road to be paved), others may not be. Particularly in models regarding the public sector, it is not always clear how to measure features of the model. For instance, when locating a desirable public facility, how do we measure accessibility of the location? We may express this feature in terms of average distance to the potential clients of this facility, or we may want to locate the facility so as to have it located within a certain distance of a majority of potential customers, or any similar measure. A particular difficulty is presented when dealing with models that include such nebulous concepts as "fairness." Take the issue of a simple speeding ticket. The penalty is supposed to hurt the speeder and thus make an impression. An obvious constraint is that the law must treat everybody equally. So whoever speeds will receive the same penalty, say $100. Such a penalty will, of course, hurt people with a small annual income a lot more than somebody who makes several hundred thousand dollars a year. Thus, while the penalty is equal, the pain is not. Alternatively, one could assess the penalty as, say 1% of the monthly net income, thus trying to distribute the pain evenly (it still does not, as a $10 penalty for somebody with a $1,000 monthly net income hurts more than $1,000 penalty for somebody with a $100,000 monthly income). This leads to ridiculous penalties, such as $50,000 for speeding and, in the final analysis, is nonsensical as it negates any incentive to earn more, as prices, fines, and other expenses are adjusted accordingly. What we want to address here is simply the difficult expressing some features of a model quantitatively.

Typically in many problems, public and private, the objective is ill-defined or fuzzy. This will require the decision maker to formulate a surrogate or proxy expression instead. For instance, the measurable criterion "profit" may be a proxy

for the "well-being of the company." Again, moving from the true objective to a proxy involves gains and losses: we lose as the proxy expression does not do exactly what we want our objective to do, but we gain by obtaining something that is measurable and thus can be included in a quantitative model. The choice of a suitable proxy expression is crucial for the applicability of the model.

Each operations research model will involve *parameters* and *variables*. While parameters are numbers that we know (or can determine), but that are not under our control, variables are numbers that we do not know (but would like to know), and which are under our direct jurisdiction. Consider a few examples. The estimated demand for a product is a parameter, while the number of units that we make is a variable. The amount of beef we put in each can of chili is under our jurisdiction and thus a variable, while the nutritional content of beans in the can is not under our control and thus a parameter. The type of truck we use for a shipment and the route the truck takes are variables, while the location of our customer's warehouse is a parameter. The purpose of the solution method is then to determine the actual values of the variables, i.e., determine the production level, the quantity of beef in a can, and the type of truck and route that a truck takes.

As far as solution techniques are concerned, we distinguish between exact and heuristic solution techniques. An exact technique will find a solution that respects all constraints included in the model and optimizes the objective specified by the decision maker. Note that the solution the exact solution method will determine is actually optimal for the model: if the model is only a very rough approximation of the problem, the solution that is labeled "optimal" will not be very useful to the decision maker, as it is optimal only for the model, not the problem. Solutions obtained by heuristic or approximation methods are typically much easier to find, but, as the name implies, are not necessarily the best (or even very good for that matter). For more details, readers are referred to Appendix A. Solution techniques come in two versions: they are either closed-form solutions or iterative algorithms. A closed form solution is essentially a formula or a set of formulas: we input the parameters and obtain values for our variables. Few models exist for which closed-form solutions exist. Most models require the use of *iterative algorithms*. (As an aside, the name algorithm derives from the Persian mathematician Al-Khwarizmi, who worked and published in algebra around 820 AD). An iterative algorithm starts with a solution, possibly a guess, the solution that is presently employed, or some other solution, and then performs a test that checks whether or not the solution satisfies certain criteria (such as feasibility or optimality). If the solution passes the test, it is accepted and the algorithm terminates, otherwise, the present solution is modified by the algorithm, a new solution (typically an improved guess) is generated, which is then again tested, and further improved, if necessary. Each loop that involves the test and an improvement is an iteration. Many modern large-scale problems require thousands, if not millions, of iterations to find a solution. This explains why the use of high-speed digital computers is crucial for the solution of today's models.

The above discussion has made frequent mention of the concept of solution. It is important to realize that a solution is a *set of instructions*. In production planning, it will tell the decision maker what quantities to produce, in diet planning it will tell the chefs what meals to prepare (e.g., in seniors' residences), and in transportation planning it will tell the dispatcher which trucks and which drivers to dispatch where, and with what loads. Associated with each solution is a value of the objective function, a value that tells the planner how much money will be made if a certain production plan is adopted, how much it will cost, if a certain meal schedule is followed, and what the consequences are if we schedule trucks and drivers in a certain way.

There are four major concerns when applying any operations research model. They are

(1) *feasibility* (can we do this?),
(2) *optimality* (is this the best we can do with what we have?)
(3) *sensitivity* (what happens, if some of the input parameters or conditions beyond our control change), and
(4) *implementability* (is the solution that we have obtained something that we can actually do?)

We will explain the first three phases in a very simple numerical

Example: A company faces a demand for its product. The magnitude of the demand is a function of the price, and both, the price p and the quantity q, are to be set by the company. It is known that the price-quantity relation is $p = 10 - 0.01q$, meaning that starting at \$10 per unit, each unit increase of the demand decreases the price all customers are willing to pay for the product by 1¢. It costs \$5 to make one unit, and general quantity-independent costs of \$500 also apply. The company wants to maximize its profit.

In order to formulate the problem, we will employ a very simple version of what is known as the *decomposition principle*. Starting with a large component of the problem that we cannot model *per se*, it subdivides this component into smaller and smaller pieces, until we are able to find expressions for it. In this example, "profit" is such a component. We first decompose profit by using its definition as revenue minus costs. Now we have to deal with these two components separately. First take revenue. Again, we decompose the entity by using the definition "revenue equals price times quantity," which leaves us with these two expressions. At this point, we are able to deal with them directly, as we know that the quantity is q and the price is $p = 10 - 0.01q$. Now consider costs. Decomposing costs, we obtain fixed and variable costs as its two components. The fixed costs in this example are known to be \$500, while the variable costs are \$5 per unit for a total of $5q$. We can then put together the profit function (the composition phase) as

$$\mathcal{P} = (10 - 0.01q)q - 5q - 500 = -.01q^2 + 5q - 500.$$

To facilitate our discussion, Figure 1.1 provides a plot of the profit function.

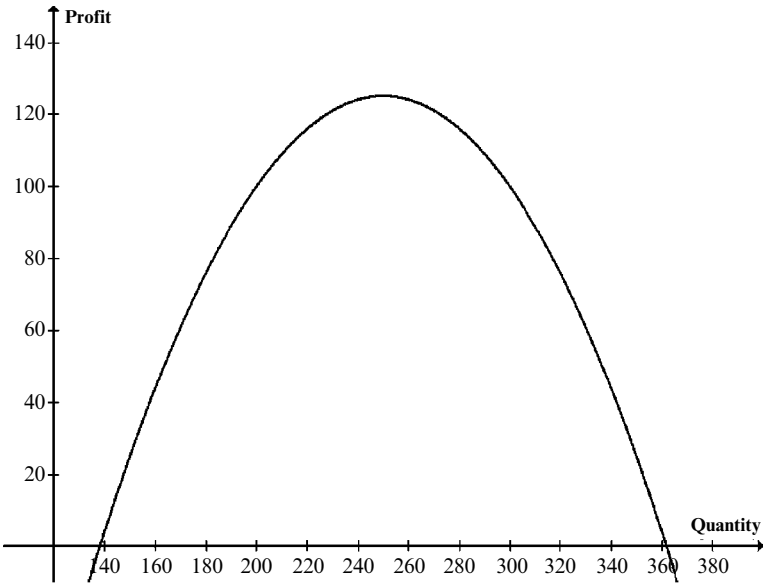


Figure 1.1

We can now explain the first three major concerns by using this example. First consider feasibility. Let us assume that the planner considers it mandatory that we are not making losses and that any solution that provides a loss is not feasible. (This is not generally true: losses obviously occur and are generally unrelated to conditions of feasibility). This leaves us with a *break-even analysis*, i.e., an analysis that determines the types of production plans that generate a nonnegative profit. This condition is $\mathcal{P} = 0$, and in Figure 1.1 we are looking for the quantities at which the profit function intersects the abscissa. In our example, this occurs at the quantities q = 138.1966 (the lowest quantity that generates a nonnegative profit) and q = 361.8034, the highest quantity that generates nonnegative profits. The prices at these two points are p = 8.6180 and 6.3820, respectively.

Next consider the issue of optimality. The quantity that generates the maximal profit is the highest point of the profit curve and it occurs at the quantity q = 250. The unit price of our product at that point equals p = \$7.50 and the profit is \$125.

Finally, let us examine what happens if some of the input parameters change. What if the fixed costs are only \$400? First we note that the optimal solution is still at $\bar{q}$ = 250. As a matter of fact, we are observing a general phenomenon: the

optimal solution will not change if we add or subtract and fixed amount to our from the objective function. The profit related to the optimal solution will, of course change: as the fixed costs have decreased by \$100, the profit at optimum has increased by \$100. The situation is different for the break-even points. They are now at $q = 100$ and 400 with the associated prices of $p = 9$ and 6, respectively. Another sensitivity analysis could ask what happens if the variable costs in the original model (we don't consider compounded changes) were to increase from \$5 to \$6. The new profit function will again be a parabola, but no point of it will be on or above the abscissa. In other words, no point with positive profit exists. The maximal profit is achieved at a quantity of $\overline{q} = 200$, where we sustain a loss of $\mathscr{P} = \$100$. As a matter of fact, this is a good example that the best possible solution (the maximal profit) is still not very good, as it does not provide a positive profit. (According to our definition of feasibility, it would not even be feasible. Actually, in this instance there are no feasible solutions).

The last sensitivity analysis—again starting from the original situation—asks what happens if the price-quantity relation changes to $p = 10 - .005q$. Again, another parabola is the profit function, and the break-even points are now at the quantities $q = 112.70$ and 887.30, while optimality is achieved at a quantity of $\overline{q} = 500$ with an associated profit of $\mathscr{P} = \$750$. In this last case, we could have guessed what would happen: as the price decreases more slowly than in the original model, there are more chances to make a positive profit, so that the break-even points will be farther apart and the optimum of the function is higher. In general, it is always a good idea to compare the results provided by a mathematical solver with those the decision maker or analyst comes up with intuitively. If optimized solution and intuition do not match, it is important to carefully check the model, as one of them will be wrong. Which one depends on the experience of the person involved and the care with which the model was formulated.

1.3 The Modeling Process

This section will outline some of the major steps that will be followed when formulating, solving, and implementing an operations research model. Clearly, each situation has its own idiosyncrasies and difficulties, but some general ideas are common to all of them. We will present the main eight steps below.

Step 1: Problem recognition. In order to build a successful model, the first step is for someone to realize that it is not "business as usual," and that it is simply no longer good enough to follow the old "we have always done it like that" and, its

sister expression in crime, “we have never done it like that.” Problem recognition does not only include the realization that things are not what they were thought to be, but also what the potential for improvement actually is. The decision maker has to keep in mind that building a model is a lengthy and expensive process that is only worth undertaking if there appears to be significant potential for improvement. This step takes a manager who is fully familiar with the actual situation in the firm and, at the very least, somewhat familiar with what operations research can do.

Step 2: Authorization to model. This step will require the analyst, who is in charge of model development to convince management of the need to produce a model for (a part of) the operation. This “sale” of potential benefits to those who eventually have to pay for it is obviously crucial. It requires very good communication skills on the part of the analyst. Often, analysts make the mistake to get lost in their technical lingo, which not only annoys decision makers, but greatly reduces the chances of obtaining permission to model. Avoiding this pitfall requires also that the analyst clearly understands the mindset and way of thinking of the decision makers.

Step 3: Model building and data collection: This is a step, in which this book can help with the model building. The first step of the modeler has to decide on the scope and the level of aggregation. Is it necessary to get into small details of the operations, or it is sufficient to adopt a macro view? This decision will have to be made in conjunction with the decision maker(s) and maybe has already been done in the previous step before the model building idea was developed and presented. Finding out what is actually important may take quite a while, but spending some time on it is definitely no waste. Once the level of aggregation has been decided upon, the modeler will determine who the relevant stakeholders are and what their objectives are. This is the time to decide on appropriate surrogate criteria. In addition to find out about from management what their objectives are, it will be necessary to learn about the constraints from the shop floor. This type of information is typically unknown in the corner offices, and it will be of tremendous benefit to the modeler and his model if such information is collected directly at the source. Some analysts went to great lengths to accomplish this. Gene Woolsey wrote about his modeling and data collection adventures in the practice journal *Interfaces* with stories that involved him getting a job as a worker in the unit he was supposed to model and seeing first hand what the actual problems were. In addition to obtaining much better data than those are typically available in offices, he also got to know the workers and their concerns. This would come as a significant benefit later on when the results of his modeling efforts would be implemented: the suggested solution would take these concerns into consideration and greatly enhance chances of the adoption of the changes, rather than resistance and passive boycotts of any suggested change. Once the data have been collected, the formal model will be built. This typically starts with a listing of the assumptions and simplifications that are made. This is important for

the decision maker later on, when the time comes to accept or reject the recommendations made by the model. It is a lot better for the decision maker to see right away that a model is not really applicable and not implement it, rather than to implement it and having to live with the (dire) consequences later, just because some assumptions did not apply in the given situation. Clearly, the assumptions should be checked periodically with the decision maker, so as to avoid wasting time. Then the decision variables are defined based on what the decision maker(s) would like to know.

Step 4: Model solution. This is a step in which the modeler's technical knowledge is required. Here, we use the appropriate computer software, and document the model properly so as to allow future users to more or less easily take over and use the model again without having to go through the entire process again. There are a few pitfalls. One of them is to use some software "just because we already have it." If the software does what we need, then it is perfectly all right to use it, but changing the model to suit existing software is a highly questionable procedure. Analysts have to keep in mind that a few thousand dollars for needed software are very little compared to the costs incurred by the model-building team and the potential benefit. Another issue to keep in mind is that modeling and model solution nowadays—in contrast to the situation some decades ago—is an interactive process. In "the olden days," computational power was severely limited and expensive. As a result, modelers tried to develop the entire model as well as possible, then had it solved and, when it came back with errors (which it always did), fixed the errors and were done. This required a lot of foresight, a lot of thought, and long and arduous searches for the errors. Nowadays, computing power is ubiquitous and cheap. A result is that modeling is typically done in an interactive fashion. A fairly small part of the model is developed first, then solved, and if there are any errors, they will be easy to find, as the model is still small. Once the analyst is happy with the modeled part, additional parts are added. The revised model is solved, and again, error detection is easy as anything wrong must be related to the new part. This process goes back and forth, until the entire model is developed.

Step 5: Model validation. In this step, the analyst will determine whether or not the solution to the model that was obtained in the previous step does make sense in the context of the real problem. If this is not the case, the model is not (yet) a faithful representation of reality and it must be changed. In such a case, the process will shuttle between Steps 3-5. It is always a good idea to determine if the solution leads to a major change as compared to the present solution (if any). If this is the case, it is rather unlikely that the new solution will be adopted by the decision makers. It is also useful at this stage to include a number of sensitivity analyses in the package prepared for the next step.

Step 6: Model presentation. In this step, the analyst(s) will "sell" the solution to management. In trying to convince them to do so, it helps a great deal if the

assumptions are clearly stated, the solution is clearly presented (and at least tentatively checked against reality in the previous step), and some alternative scenarios are also presented. As a matter of fact, in all cases other than the lowest operational level, on which solutions are more or less automatically implemented, there are decision makers, whose job is to make decisions, not to accept or reject decisions from analysts. This means that the main function of operations research is not so much decision making, but preparing the decision. Presenting some workable decision alternatives is usually a good idea. This is a crucial step, which does not only decide about the future of the model, but possibly the future of the modelers themselves: no firm will keep employees whom it does not see contributing to the benefit of the organization, and producing models that are not used is no benefit. As a matter of fact, among the models that decision makers actually asked to be built, only a surprisingly small portion were ever implemented. This is highly detrimental to the firm and their employees in particular and the profession as a whole.

Step 7: Implementation. Given the acceptance by the decision makers (most likely with some modifications of the solution), the task is now to translate the model recommendations into practice. At this point it will help a great deal if those who have to live with the recommendation and the changes—the employees and workers—will accept the solution rather than sabotage it. If the modeler created goodwill in Step 3 of this procedure, listened to the concerns of those involved, and included them in the model as far as possible, chances of acceptance are much enhanced. In addition, as the solution of the model is implemented, it is crucial to monitor the implementation each step of the way, so that it is possible to adjust the solution in case some unexpected changes occur.

Step 8: Monitoring and Control. This phase of the process is largely overlooked. It includes the timely comparison of the plan and reality. Reality changes, and the plan should be adjusted accordingly. As the famous saying in the United States Marine Corps goes, “improvise, adapt, overcome.” Furthermore, the more frequently adjustments can and are being made, the less dramatic they will have to be. As an example, assume that an individual has planned his budget for the upcoming year. Suppose that $1,200 have been set aside for the purpose of entertainment. If the individual monitors the situation monthly, he may find by the end of January that he has already spent $500 on entertainment purposes. This leaves $700 for the remainder of the year, or $63.64 for each of the remaining eleven months. This is considerably less than the $100 that were planned for each month, but is still not too dramatic. Monitoring only each second month, the individual has not noticed that he spends way too much money and behaves in February in the same way as in January by spending another $500. Checking the situation by the end of February, there are only $200 for each of the remaining months, or $20 for each months. This is a much more severe decrease from the originally planned $100 per month and will be much more difficult to adhere to.

2 Linear Programming

This chapter will introduce linear programming, one of the most powerful tools in operations research. We first provide a short account of the history of the field, followed by a discussion of the main assumptions and some features of linear programming. Thus equipped, we then venture into some of the many applications that can be modeled with linear programming. This is followed by a discussion of the underlying graphical concepts and a discussion of the interpretation of the solution with many examples of sensitivity analyses. Each of the sections in this chapter is really a chapter in its own right. We have kept them under the umbrella of the chapter "Linear Programming" so as to emphasize that they belong together rather than being separate entities.

2.1 Introduction to Linear Programming

The purpose of this section is to provide a short introduction to linear programming problems. We first present a short historical account of the topic, followed by the main assumptions of linear programming problems, and finally some details about the optimization problems under discussion.

As already discussed in Chapter 1, linear programming problems were first described by George Bernard Dantzig in 1947. His findings that included his "simplex method for linear programming" were presented in 1949 at a conference for research in economics, and some of the papers, including Dantzig's work, were published in 1951 in a volume edited by (later Nobel prize laureate) Tjalling C. Koopmans. Much work was done in the early years. The first programmed solution code based on Dantzig's simplex method was already developed in 1951, many applications were first discussed, among them the blending of aviation gasolines and trim loss problems. The dual simplex method by Lemke and the Hungarian method for assignment problems were also described and published in the 1950s. In 1963, Dantzig's book "Linear Programming and Extensions" was published, an "instant classic." The year 1979 saw the publication of a paper by Khachian, whose "ellipsoid method" made quite a stir. However, while it has some remarkable theoretical properties, it failed to perform well in practice and is not in use nowadays. On the

H.A. Eiselt and C.-L. Sandblom, *Operations Research: A Model-Based Approach*,
DOI 10.1007/978-3-642-10326-1_2, © Springer-Verlag Berlin Heidelberg 2010

other hand, Karmarkar's interior point method, first described in 1984, has slowly made some inroads. To this day, though, Dantzig's simplex method is the method of choice for the vast majority of all linear programming problems that are solved.

In order to structure our discussion, first consider a general mathematical programming problem. It can be written as the problem

P: Max $z = f(x)$
s.t. $g(x)\ R\ b$,

where P stands for problem (sometimes we have multiple problems under consideration, in which case we will write P_1, P_2, and so forth), $f(x)$ is the objective function, which is to be maximized and z is the objective function value (e.g. profit, market share, sales, or cost) associated with the present solution x. This objective is to be optimized subject to (or s.t. for short) the constraints $g(x)\ R\ b$. We will come to them and their meaning again later.

Linear programming problems are special cases of the above formulation. In particular, we have three fundamental assumptions in linear programming. They are:

(1) *Deterministic property*,
(2) *divisibility*, and
(3) *linearity*.

Consider first the deterministic property. Simply put, it requires that all parameters are assumed to be known with certainty. (By the way, the antonym of deterministic is *probabilistic* or *stochastic*). While this assumption appears reasonable in some instances, it is not in others. Consider these examples. While we know exactly how many machines we have for the processing of semi-finished products, these machines may fail unexpectedly, making their actual capacities probabilistic. Similarly, we know the magnitude of contracted sales, but we have only rough estimates concerning next month's additional demand. Given that much of the world is probabilistic, how can we possibly apply linear programming without making simplifications that may be so major so as to render the results unusable? (Remember the old adage: "if a model does not fit the problem, don't use it.") In general, there are two ways to circumvent the problem. One would be to resort to techniques that do not make that assumption—such as stochastic programming, a very complex field for which the available software is much less powerful—or to circumvent the difficulty by using sensitivity analyses. This is the procedure of choice for many decision makers. Suppose we have estimated next month's demand to be 500 units, but we are not sure about it. One way to handle this is to solve the problem with a demand of 500 units. Once the optimal solution is known, we attempt to find out what happens if the demand were to change to values other than 500. (Notice the words "what – if" that always indicates sensitivity analyses). For instance, we can change the demand to, say 490, resolve the problem and find out what happens to our solution. This process can be

repeated to the relevant values of the demand. Once this has been accomplished, the decision maker will have a clear idea what the effects on different levels of demand are on the solution of the problem. The limitations of this approach are obvious: the method is valid if only a small number of uncertain parameters exist, but it will drown the decision maker in massive amounts of data, if much of the information in the model is uncertain.

Next consider the issue of divisibility. Divisibility is ensured, if we allow the variables in the model to be any real number (even though we will restrict ourselves to rational numbers most of the time). Most importantly, divisibility allows variables to be noninteger. While many instances do not allow it, this assumption is typically less of a problem that we may first think. Suppose that our model includes a variable that expresses the number of cans of corn that we make and sell. It is obvious that this number must be integer, as partial cans cannot be sold. The main question is, though: who cares if we make 1,345,827 or 1,345,826 cans? Many of the numbers in our model will be approximations anyway, so that it is completely irrelevant if the number of cans in the optimal solution is integer or not. Simple rounding will solve the problem. On the other hand, suppose that the model under consideration includes a variable that expresses the number of houses to be built in a subdivision. Simply rounding a noninteger solution up or down may result in solutions that may not be as good as possible (or, depending on the constraints, not even feasible). If integrality of a variable is essential, users are well-advised to resort to techniques from integer programming, see Chapter 4 of this volume.

Finally the issue of linearity. (For a discussion of the issue of linearity, see Appendix C of this book). The assumption in linear programming is that all expressions, the objective function as well as all constraints, are linear. One of the underlying assumptions that leads to linearity is proportionality. As an example, if we purchase one pound of potatoes for, say, 50¢, then proportionality means that two lbs will cost me \$1, three lbs will cost \$1.50, and so forth. So far, our costs are proportional to the quantity and the cost function is $0.5x$ with \$0.5 the per-unit price of the potatoes and x denoting the quantities of potatoes we purchase. This is part of a linear function. However, if the quantities we purchase become large, the merchant may offer us a rebate, which will make the function nonlinear (or piecewise linear).

Having stated all of the main assumptions we can now focus on the individual components of linear programming problems. First consider the objective function. In all of mathematical programming, objective functions either maximize a function or they minimize it. Often, the objective expresses the wishes of the decision maker in monetary terms, but nonmonetary objectives are possible and common in applications in engineering and in the public sector. Note that if the objective is Max $z = f(x)$, it can alternatively and equivalently be written as Min $-z = -f(x)$. Suppose that originally, the idea was to maximize profit, then the alternative objective is to minimize losses. In other words, each maximization

function can be rewritten as a minimization function and vice versa. This eliminates the need for separate methods or treatment of problems with minimization and maximization objective.

Consider now the constraints. Rather than writing them in the very general form $g(x)\ R\ b$ that is often used in mathematical programming, we will write all constraints as *LHS R RHS*. What this means is that the left hand side *LHS* is in some desired relation to the right-hand side *RHS*. First, as relations linear programming accepts $\leq$, $=$, and $\geq$ relations. This relation compares a feature of the solution at hand with a proscribed standard. For instance, the number of pallets of corn shipped out of a warehouse (the reality, what "is") on the left-hand side is compared to the quantity that our customers actually wanted (the stipulated target, the "should") on the right-hand side. Similarly, we can compare our actual expenditures on the left-hand side with the targeted expenditures (i.e., our budget) on the right-hand side. Typically, constraints are formulated so as to have a linear function on the left-hand side and a single parameter on the right-hand side.

We will defer our discussion of the many facets of constraints to Section 2.1 of this book, where a variety of applications of linear programming are discussed. At this point, we will only mention a few general principles that relate to constraints. As already discussed in Chapter 1, constraints are "hard" in the sense that the linear programming solver will return a "there exists no feasible solution" message in case the set of constraints does not allow a solution to be found. This means that constraints are absolute and must be satisfied exactly. In other words, given that we have \$100, any plan that includes expenditures of say \$100.01 will not be considered as feasible. In reality, often constraints are much softer. This will have to be considered in the formulation, for instance by formulating constraints more loosely, e.g., consider the possibility of taking out a loan to increase the budget, or by allowing solutions that do not fully satisfy a customer's demand. Another important advice for modeling is to avoid equations whenever possible. Linear programming is well equipped to handle equations from a technical point of view, but equations are very restrictive and often lead to infeasibilities. We will return to this subject when we discuss applications and the graphical solution method.

In order to more fully explain the structure and features of constraints, we consider a very small numerical example. Suppose that we can manufacture digital picture frames for the domestic market and frames for export. The assembly of a domestic frame takes ten seconds each, while an export frame takes 12 seconds. The capacity of the machine on which both frames are assembled is 20 hours (or 72,000 seconds). Our distributors has placed orders for 2,500 domestic frames and 3,000 export frames, but is prepared to take more. Each domestic frame nets us \$40, while each export frame contributes \$45 to our overall profit.

Defining x_1 and x_2 as the number of domestic and export frames made and sold, respectively, we can formulate the linear programming problem as

$$
\begin{aligned}
\text{P: Max } z &= 40x_1 + 45x_2 \\
\text{s.t.} \quad 10x_1 &+ 12x_2 \leq 72{,}000 \\
x_1 &\geq 2{,}500 \\
x_2 &\geq 3{,}000
\end{aligned}
$$

Whenever an inequality of either "≤" or "≥" type is inputted, the solver will automatically add an additional variable to the equation. These variables have an important interpretation, which is why we will explain them here. In particular, whenever we have a constraint of type $LHS \leq RHS$, the solver automatically adds a *slack variable* S to the left-hand side of the constraint and then uses the constraint $LHS + S = RHS$ with $S \geq 0$. As an example of what a slack variable means and why this procedure is valid, consider a simple budget constraint. In such a constraint, the left-hand side expresses the amount of money we spend, while the right-hand side specifies the amount of money that is available, so that the constraint reads "the amount of money spent must be less or equal than the amount of money available." The slack variable is then the difference between left- and right-hand side. In our budget constraint, the rewritten constraint then states that the amount of money used (the original *LHS*) plus the amount of money unused (the slack *S*) equals the amount of money available (the original *RHS*).

A similar procedure is used for inequalities of the "≥" type. Here, the original constraint $LHS \geq RHS$ is transformed into an equation by subtracting a *surplus or excess variable* E from the left-hand side so as to arrive at the reformulated constraint $LHS - E = RHS$. For instance, in a production requirement, where the left-hand side expresses the actual production, while the right-hand side value specifies the smallest quantity that must be made, the excess variable specifies the amount by which the present solution exceeds the requirement.

Consider now the above numerical example and suppose we have obtained the solution $x_1 = 2{,}800$ and $x_2 = 3{,}200$. First of all, this indicates that we have decided to manufacture 2,800 frames for the domestic and 3,200 frames for the export market. By plugging these values into the objective function, we can determine the profit level for this solution as $z = 40(2{,}800) + 45(3{,}200) = \$256{,}000$. We can also check if this solution is feasible: the plan requires $10(2{,}800) + 12(3{,}200) = 66{,}400$ seconds to make, which is less than the available assembly time. As a matter of fact, given that we have 72,000 seconds on the assembly machine and this solution uses 66,400 seconds, there is a slack capacity of $72{,}000 - 66{,}400 = 5{,}600$ seconds. The next two demand constraints are also satisfied: since we make 2,800, this is more than we need to (actually, an excess of $2{,}800 - 2{,}500 = 300$ units); similarly, 3,200 units for the export market are $3{,}200 - 3{,}00 = 200$ units in excess of the requirement.

As an aside, the solution discussed in the previous paragraph is not optimal. At optimum, we make 3,600 units for the domestic market and 3,000 units for the export market. This results in zero slack capacity for the assembly machine and a

surplus of 1,100 units for the domestic market and a zero surplus for the export market. The profit at optimum is $279,000.

2.2 Applications of Linear Programming

This section presents a variety of linear programming applications. Each of these applications is a prototype, in the sense that "real" applications in practice will build upon these formulations by adding lots of "bells and whistles." However, the major use of learning about these bare-bones formulations is to understand the type of formulations they present and the way the variables are defined, which is typical for the type of application.

The following subsections describe in detail how some scenarios can be modeled. They may be simplistic, but the models presented here include the most important features of the application under consideration.

2.2.1 Production Planning

The formulation that we present in this section is a very basic prototype of production planning models. It is probably the simplest possible problem to be formulated, and the purpose to present it at this point is to introduce some of the basic ideas of modeling. In this context, suppose that there are three products we can manufacture. For simplicity, we will simply refer to them as P_1, P_2, and P_3, respectively. The three products sell for $20, $15, and $17, respectively. These products are manufactured on two machines M_1 and M_2. Each of the two products has to be processed on both machines. At this level, the order in which to process the products on the machines is irrelevant. The two machines have capacities of 8 and 9 hours, respectively. After that time, the machines will require regular maintenance, a task for which they have to be shut off and are no longer available. The processing times of the three products on the two machines (in minutes per quantity unit) are shown in Table 2.2.1.

Table 2.2.1: Processing times of the products on the machines

	P_1	P_2	P_3
M_1	3	5	4
M_2	6	1	3

It is important to understand that the machine capacities and their usage are meant in the following way. Suppose there is a clock attached to the machine that shows the amount of time that is still available on the machine before the next scheduled maintenance. Initially, the clock on the first machine shows 9 hours or 540 minutes. Suppose now that four units of P_2 are processed on M_1. Given that the processing time per unit is 5 minutes, the available time decreases by twenty minutes from 540 to 520. This process continues until one of the machines has no capacity left.

This process is considered automatically in the formulation, we present it here so as to stress the way time is managed here. It is often misunderstood that a capacity of, say, 8 hours means a workday and the objective is to plan what product is processed at what time on which machine. This type of micro planning is discussed in detail in Chapter 8 of this volume.

The production costs are \$120 per hour of machine time on M_1, and \$90 per hour of machine time on M_2. In other words, operating time costs \$2 and \$1.50 per minute on the two respective machines. Given the operating times, the processing costs of the three machines on M_1 are \$6, \$10, and \$8, respectively, while the operating costs on M_2 are \$9, \$1.50, and \$4.50, respectively. This results in overall unit processing costs of \$15, \$11.50, and \$12.50 for each of the three products, respectively. Considering the selling prices, we have profit contributions of \$5.00, \$3.50, and \$4.50 per unit of each of the three products.

The problem can then formulated so as to maximize the overall profit, while respecting the capacity constraints of the two machines. In order to formulate the problem, we first need to define the variables. In this example, we would like to know the number of products that we will manufacture. How much time we will use on the two machines in the process does not have to be defined as a variable, it is a direct consequence of how many products we will make.

Defining x_1, x_2, and x_3 the respective numbers of P_1, P_2, and P_3 that we will manufacture and sell, we can then start to formulate the objective function. Knowing that the objective will maximize the profit, which is the sum of profits that derive from the making and selling of products, we consider one such term at a time. For instance, making and selling one unit of P_1 nets \$5, and we have now decided to make x_1 units of it. This means that the profit that derives from the making and selling of P_1 will be $5x_1$, Similar expressions can be derived for the other two products, resulting in the objective function shown below in the formulation.

Next, consider the constraints. We need to formulate one constraint for each machine. Each of these constraints will state that the actual usage of the resource called processing time does not exceed the processing time that is available. The time available is known, so the right-hand side of the constraints is easy. In order to be more specific, we will deal with the first machine and formulate the actual time that is used on M_1. First, we note that the time will be consumed by the making of the three products. Consider one such product at a time, say P_1. We know that it takes 3 minutes to process one unit of P_1 on M_1, and we have decided to make x_1 units of P_1. This means that the processing time on M_1 that is used for the making of P_1 is $3x_1$. Similarly, we will use a total of $5x_2$ minutes on M_1 to make all the units of P_2 that we will manufacture, and we need $4x_3$ minutes on M_1 to make all of P_3. The sum of these processing times now must be less than or equal to the available time of 540 minutes. This is the capacity constraint for M_1.

The capacity constraint for the second machine is formulated in similar fashion. Finally, we have to ensure that we are making nonnegative numbers of the products—after all, we are in the business of manufacturing and selling the products, rather than buying them (which is what negative values of the variables would indicate in this context). The formulation can then be written as the problem

$$\begin{aligned} \text{Max } z = \; & 5x_1 + 3.5x_2 + 4.5x_3 \\ \text{s.t.} \quad & 3x_1 + 5x_2 + 4x_3 \leq 540 \\ & 6x_1 + 1x_2 + 3x_3 \leq 480 \\ & x_1, \quad x_2, \quad x_3 \geq 0. \end{aligned}$$

After solving the problem, the optimal solution indicates that the decision maker should make and sell 20 units of P_1, no units of P_2, and 120 units of P_3. The associated total profit is \$640.

2.2.2 Diet Problems

The diet problem is not only among the first linear programming problems to be solved, but it is arguably also the most intuitive model. As a matter of fact, the problem was studied in 1939 by the (later Nobel laureate) George Stigler in the context of determining a cost-minimal nutritious food for the armed forces. Incidentally, he included 77 different foods in his model. His solution, while not optimized, was found to be quite close to the optimal solution for his data. An interesting account of the history of the diet problem is found in Garner Garille and Gass (1981).

In general, two versions of the problem can be thought of. They are roughly equivalent to the fundamental economic principle, applied to the determination of a diet. The two criteria are cost and nutritional value, where cost is an input factor, while nutritional value is an output. Thus we can either try to minimize the cost (the input), while guaranteeing at least a certain and predetermined nutritional value (the output), or we can attempt to maximize the nutritional value (the output), while using no more than a prespecified dollar value (the input). In order to determine which of the two versions is more suitable, consider this. In the latter approach, the constraint is a simple resource constraint that states that the total amount of money spent on food cannot exceed the decision maker's budget. However, the objective function is more complex. It is stated to maximize the nutritional value. The problem with this is that the measure "nutritional value" is not a simple number, it is what is usually referred to as multidimensional measure, as it comprises a large number of measures: protein, carbohydrates, fats, vitamins, minerals, etc. And it is obviously not meaningful to simply add those different units together so as to obtain a single one-dimensional measure.

On the other hand, the former version of the problem is much easier to handle. The objective is a simple (one-dimensional) cost minimization, in which the "nutritional content" is relegated to the constraints. And this is where the requirements can be

handled quite easily: for each nutritional component, we can formulate one or two constraints, an upper and/or a lower bound. As a simple numerical example, consider three foodstuffs, e.g., hamburger, fries, and cheesecake (which indicates that we are using the term "diet" in the widest possible sense). As far as nutrients are concerned, we consider only calories and fat in our example. Table 2.2.2 shows the nutritional contents of the foodstuffs, the nutritional requirements, and the prices per serving of the foods.

Table 2.2.2: Input data for the sample problem

	Hamburger 3½ oz	Medium fries 4 oz	Cheesecake 2.8 oz	Nutritional requirement
Calories	250	380	257	[1,800; 2,200]
Fat	13%	31%	28%	≤ 100%
Cost per serving	$1.59	$2.19	$2.99	

Note that our numerical example expresses the calories in terms of the usual kCal and we have an upper and a lower bound for it. (Note that in practice, the recommended caloric intake depends on gender and age of an individual as well as other factors). On the other hand, the fat content is expressed in terms of the recommended daily requirements and it is an upper bound.

The problem in the cost minimization version can then be formulated as follows. First, we define the variables. As usual, the variables are defined depending on what the decision maker would like to know, but does not at the moment. In this model, the decision maker's goal is to determine the quantities of the individual foods that are to be included in the daily diet, so that we will define variables x_1, x_2, and x_3 as the quantities of hamburgers, medium fries, and cheesecakes in the diet. As long as we use well-defined "servings" as units, and stick to the same units for each food, there is no problem inadvertently adding apples and oranges, or hamburgers and fries. What we are adding are their nutritional contents, as we show in this example.

First consider the objective function. The general idea is to minimize the costs of all foodstuffs in the diet. Consider one of these foods at a time. As shown in Table 2.2.2, each hamburger costs $1.59. Since there will be x_1 hamburgers in the diet, the total costs incurred by hamburgers in the diet are $1.59x_1$. Similarly, the costs that can be attributed to fries and cheesecake are $2.19x_2$ and $2.99x_3$, respectively, so that the objective function is the sum of these three terms.

Let us now formulate the constraints. In doing so, we consider one requirement at a time. First, we focus on the lower bound of the constraints. What we would like to express is that the caloric content of the diet should be at least 1,800. The calories in the diet derive from the three foods. the number of calories in the diet that are from hamburgers are $250\ x_1$, the number of calories from fries are $380x_2$, and the calories from cheesecake are $257x_3$. Adding up these three expressions results in

the total number of calories that we actually have in the diet. This number should not fall short of 1,800 (the first constraint), and it should also not exceed 2,200 (the second constraint). Note that both of these constraints have the same left-hand side.

The constraint that regulates the fat content of the diet is constructed similarly. Overall, the constraint should state that the content of fat in the diet should not exceed 100% of the recommended daily value. Where does the fat in the diet come from? In this example, it derives from hamburgers (for a total of $13x_1$), fries (for a total of $31x_2$), and cheesecake (for a total of $28x_3$). This total fat content should then not exceed 100%.

In summary, the problem can then be formulated as follows.

$$\text{P: Min } z = 1.59x_1 + 2.19x_2 + 2.99x_3$$

$$\begin{aligned}
\text{s.t. } & 250x_1 + 380x_2 + 257x_3 \geq 1{,}800 && \text{(calories, lower bound)}\\
& 250x_1 + 380x_2 + 257x_3 \leq 2{,}200 && \text{(calories, upper bound)}\\
& 13x_1 + 31x_2 + 28x_3 \leq 100 && \text{(fat, upper bound)}\\
& x_1, \quad x_2, \quad x_3 \geq 0. && \text{(nonnegativity constraints)}
\end{aligned}$$

In addition to the lower and upper bounds on the caloric intake and the upper bound on the fat content of the diet, we include the nonnegativity constraints on the quantities of foods included in the diet, as the idea is to eat rather than regurgitate. Incidentally, the solution to this formulation specifies that the planner eat 6⅓ hamburgers, ½ serving of fries, and no cheesecake. Doing so will provide 100% of the allowable fat intake, generate 1,800 calories, and cost $11.32.

While this solution may be optimal with respect to the problem as specified above, it is clearly riddled with a number of problems. The number of hamburgers in the diet is obviously much too high for all but the most determined junk food junkies. It will require additional constraints which are added in the loop that attempts to reconcile the solution of the model with the real problem of finding a cost-effective diet. In order to illustrate the modeling process, we use a somewhat larger problem with real data. The data of the eight foods included and the eleven nutrients is found in Table 2.2.3.

As far as the daily nutritional requirements are concerned, we have identified the following parameters. The diet should include

- between 1,800 and 2,200 calories,
- no more than 65g of fat,
- no more than 300mg of cholesterol,
- no more than 2,400mg of sodium,
- at least 300g of carbohydrates,
- at least 25g of fiber,
- at least 50g or protein, and
- at least 100% of the recommended daily allowance of vitamins A and C, calcium, and iron.

Table 2.2.3: Nutritional contents of foods and their prices

	Pasta	Tomato juice	Clam chowder	Medium beef chuck	Milk	Orange juice	Apple	Potato chips
Calories	300	60	220	259	110	132	55	152
Fat	1g	0g	13g	16.3g	2.5g	0g	0.22g	9.8g
Cholesterol	0mg	0mg	5mg	89mg	10mg	0mg	0mg	0mg
Sodium	1mg	650mg	790mg	95mg	120mg	5mg	1.1mg	168.4mg
Carbohydrates	63g	12g	19g	20g	12g	33.4g	14.6g	15g
Fiber	3g	3g	2g	0g	0g	0g	2.5g	1.3g
Protein	11g	2g	5g	26.1g	9g	0.5g	0.3g	2g
Vitamin A	0%	8%	2%	1%	10%	2%	1%	0%
Vitamin C	0%	30%	2%	0%	0%	62%	8%	15%
Calcium	2%	2%	2%	1%	30%	0%	1%	1%
Iron	20%	15%	8%	17%	0%	2%	1%	3%
Price per serving	19¢ per 3 oz	56¢ per 10 fl.oz.	90¢ per 8.8 fl.oz.	82¢ per 3 1/2 oz	51¢ per cup	53¢ per 8.8 fl.oz.	37¢ each	32¢ per 1oz

We first define variables $x_1, \ldots, x_8$ for the number of servings of the eight foods outlined in Table 2.2.3. Following the ideas developed above for the small problem, we can now formulate the diet problem. the formulation is as follows.

$$\text{P: Min } z = .19x_1 + .56x_2 + .90x_3 + .82x_4 + .51x_5 + .53x_6 + .37x_7 + .32x_8$$

$$\text{s.t. } 300x_1 + 60x_2 + 220x_3 + 259x_4 + 110x_5 + 132x_6 + 55x_7 + 152x_8 \geq 1{,}800$$
(Calories, lower bound)
$$300x_1 + 60x_2 + 220x_3 + 259x_4 + 110x_5 + 132x_6 + 55x_7 + 152x_8 \leq 2{,}200$$
(Calories, upper bound)
$$1x_1 + 13x_3 + 16.3x_4 + 2.5x_5 + 0.22x_7 + 9.8x_8 \leq 65$$
(Fat)
$$5x_3 + 89x_4 + 10x_5 \leq 300$$
(Cholesterol)
$$1x_1 + 650x_2 + 790x_3 + 95x_4 + 120x_5 + 5x_6 + 1.1x_7 + 168.4x_8 \leq 2{,}400$$
(Sodium)
$$63x_1 + 12x_2 + 19x_3 + 20x_4 + 12x_5 + 33.4x_6 + 14.6x_7 + 15x_8 \geq 300$$
(Carbohydrates)
$$3x_1 + 3x_2 + 2x_3 + 2.5x_7 + 1.3x_8 \geq 25$$
(Fiber)
$$11x_1 + 2x_2 + 5x_3 + 26.1x_4 + 9x_5 + 0.5x_6 + 0.3x_7 + 2x_8 \geq 50$$
(Protein)
$$8x_2 + 2x_3 + 1x_4 + 10x_5 + 2x_6 + 1x_7 \geq 100$$
(Vitamin A)
$$30x_2 + 2x_3 + 62x_6 + 8x_7 + 15x_8 \geq 100$$
(Vitamin C)
$$2x_1 + 2x_2 + 2x_3 + 1x_4 + 30x_5 + 1x_7 + 1x_8 \geq 100$$
(Calcium)
$$20x_1 + 15x_2 + 8x_3 + 17x_4 + 2x_6 + 1x_7 + 3x_8 \geq 100$$
(Iron)
$$x_1, x_2, \ldots, x_8 \geq 0$$
(Nonnegativity constraints)

In the subsequent discussion, we will refer to the solutions and their respective nutritional contents shown in Table 2.2.4a and Table 2.2.4b (where entries in are shown in boldface, if the nutritional content of a diet has reached its bound). Solving the above model results in the original solution "0." The most prominent features of the solution is that other than a modest amount of pasta, the diet includes only liquids, most prominently in excess of 9 servings of milk. The decision maker may want to limit the milk intake to more reasonable levels, so that an upper bound of four servings of milk is added, i.e., the constraint $x_5 \leq 4$. It turns out that this constraint is so strong that the model has no more feasible solution. In order to restore feasibility, the decision maker has now decided to add additional foods, most prominently yoghurt, bread, and margarine (whose quantities are denoted by the new variables x_9, x_{10}, and x_{11}, respectively). The prices and nutritional contents of the foods are shown in Table 2.2.5. Solving the

Table 2.2.4a: solutions in the modeling process

Foodstuff	Original Solution No. "0"	≤ 4 milk	Solution 1	≤ 6 bread Sln 2	≤ 4 margarine Sln 3	Add eggs Sln 4	Calories ≤ 2,000 Solution 4a	Cholesterol ≤ 250 Solution 4b
Pasta	2.58	No	0.1	1.46	0	0	0	0
Tomato juice	1.95	feasible	0	0.91	0	0	0	0
Clam chowder	0	solution.	0	0	0	0	0	0
Beef chuck	0.09	Introduce	0	0	0.38	0.2	0.32	0.36
Milk	9.3	yoghurt,	3.87	3.45	4	4.0	4	4
Orange Juice	0.08	bread,	1.33	0.87	5.47	3.47	2.76	3.2
Apple	4.6	margarine	2.23	2.35	8.68	6.0	7.61	8.2
Potato chips	0		0	0	0	0	0	0
Yoghurt			0	0	0	0	0	0
Bread			9.57	6	1.65	5	2.99	2.2
Margarine			5.65	6.05	4	4	4	4
Eggs						1.14	1.09	0.84
Cost	$8.14		$4.62	$4.70	$8.81	$7.12	$7.26	$7.63

problem with the new foods results in solution "1," shown again in Tables 2.2.4a and 2.2.4b.

Table 2.2.4b: Nutritional achievements of the solutions

	Original solution "0"	Solution "1"	Solution "2"	Solution "3"	Solution "4"	Solution "4a"	Solution "4b"
Calories $\in$ [1,800; 2,200]	**1,800**	**2,200**	**2.200**	**2,200**	**2,200**	**2,000**	**2,000**
Fat ≤ 65	28	**65**	**65**	52	57	58	56
Cholesterol ≤ 300	101	39	35	74	**300**	**300**	**250**
Sodium ≤ 2,400	**2,400**	**2,400**	**2,400**	1,097	1,674	1,358	1,231
Carbs ≥ 300	369	340	340	401	366	324	332
Fiber ≥ 25	**25**	**25**	**25**	**25**	**25**	**25**	**25**
Protein ≥ 50	120	75	74	58	72	67	64
Vitamin A ≥ 100%	**100**	**100**	**100**	**100**	**100**	**100**	**100**
Vitamin C ≥ 100%	**100**	**100**	**100**	409	263	232	264
Calcium ≥ 100%	293	118	111	129	130	131	131
Iron ≥ 100%	**100**	103	107	**100**	**100**	**100**	**100**

We first notice that the price of the diet has decreased dramatically, a result of new inexpensive foods that have been introduced into the problem. (As usual, while added constraints limit the choices and increase the cost of the solution, added variables represent added opportunities that may reduce the price). The solution now includes significant amounts of the new foods bread and margarine. The nutritional content of the new solution also differs quite dramatically from that of the previous solution: the caloric content is now at the upper rather than the lower bound, the fat content is also at the upper bound, while sodium is still at the upper bound, and the nutrients fiber, vitamin A and vitamin C are at the lower bounds.

The decision maker may feel that the present solution includes too much bread, so that a constraint is added that limits the bread content of the diet to no more than six slices, i.e., $x_{10} \leq 6$. The result is Solution "2." This solution is marginally more expensive than its predecessor, it features significantly more pasta, and in general

appears more palatable. however, the margarine content is quite high, so that in the next step, the decision maker adds a requirement that limits the margarine content to no more than four servings, i.e., $x_{11} \leq 4$. The result is Solution "3." This requirement causes the price of the diet to almost double (which poses the question whether or not the requirement is really *that* crucial). Furthermore, the diet now consists mostly of milk, orange juice, and apples. It is hardly surprising that the vitamin C content of the diet has quadrupled as compared to the previous solution. Also, fat is no longer a binding constraint and the sodium content is cut in half. However, the diet is still barely satisfying the fiber content, and its vitamin A content is also at its lower bound.

Table 2.2.5: Additional foods

	Yoghurt	Bread	Becel	Eggs
Calories	160	110	70	78
Fat	2.5g	1g	8g	5.3g
Cholesterol	10mg	0mg	0mg	212mg
Sodium	75mg	160mg	70mg	62mg
Carbs	20g	22g	0	0.6g
Fiber	0g	2g	0	0g
Protein	6g	4g	0g	6.3g
Vitamin A	2%	0%	10%	6%
Vitamin C	2%	0%	0%	0%
Calcium	20%	0%	0%	3%
Iron	2%	10%	0%	3%
Price per serving	56¢ per ¾ cup	8.6¢ per slice	5¢ per 2 teaspoons	20¢ each

In order to address some of the nutritional issues, the decision maker has decided to add eggs as an additional food. Their price and nutritional content are shown in Figure 4. Solving the expanded model results in Solution "4." Now the diet includes again more bread and margarine, and more moderate amounts of milk, orange juice, and apples. Also, the price has dropped to $7.12.

In principle, the decision maker is happy with the diet as it has been determined at this point. However, it may be desirable to find out the effects that result from the changes of some of the requirements. For instance, what happens if the caloric intake is restricted to no more than 2,000 rather than 2,200? Solving the revised model results in Solution 4a. We notice a very modest price increase, of a diet that now includes less orange juice, more apples, and significantly less bread. We are now at the upper allowable limit for cholesterol, while fiber, vitamin A and iron are stubbornly clinging to their respective lower bounds. (A good idea at this point would be to introduce additional foods that are rich in these nutrients).

Instead, the decision maker may wish to reduce the cholesterol in the diet from its present upper bound of 300 to 250. Resolving the problem results again in a fairly modest price increase and a solution that is similar to the previous diet, except that

now the quantity of apples has reached again unreasonable levels. We terminate our discussion at this point, which is not to suggest that the present diet is reasonable: the purpose of this section was to introduce the reader to the modeling process that repeatedly revises the model based on the present solution.

2.2.3 Allocation Problems

Allocation problems are one of the most prominent areas of application in linear programming. All models in this class have in common that they deal with the allocation of scarce resources to (economic) activities. At times, more than one scarce resource exists, in which case the modeler can choose any one of them as the basis for the mathematical formulation. This will be further elaborated upon below. Due to the many different types of allocation problems, we will present two such scenarios below.

First consider an investment allocation problem. Problems of this nature were first formulated by Markowitz in the early 1950s as nonlinear optimization problems. Here, we will discuss a linear version of the problem. In this problem, the decision maker has to decide how much of the scarce resource (money) to allocate to different types of investments. As we will see below, this observation already indicates how to define the variables.

In our numerical example, the investor has $300,000 that can be invested. In addition to the money at hand, it is possible to borrow up to $100,000 at 12% interest. This money can be used for leveraging (borrow to invest). The investor has narrowed down the choices to six alternatives, shown in Table 2.2.6. The table also shows the expected annual interest or dividend for the investment alternatives, the expected annual increase of the value of the investment, and an indication of the risk of the investment (per dollar).

Table 2.2.6: Types of investments and their features

Investment type	Expected annual interest/dividend	Expected annual increase in value	Average risk per dollar
Real estate	0%	18%	20
Silver	0%	10%	12
Savings account	2%	0	1
Blue chip stocks	3%	6%	7
Bonds	4%	0%	3
Hi-tech stocks	0%	20%	30

We will consider two versions of the problem: the first version attempts to maximize the expected value of the assets at the end of the planning period (one year), while the second version minimizes the total risk of the investment.

First consider version 1. The value of the assets after one year equals today's value of the investment plus the expected interest or dividend plus the expected change in value within a year minus the amount of money that was borrowed (principal and interest). In addition to the restricted availability of money already mentioned above, the decision maker faces the following constraints:

- The expected value of assets (exclusive interest) at the end of the planning period should be at least 7% higher than at the beginning,
- invest at least 50% of all the money invested in stocks and bonds combined,
- invest no more than 20% of total amount available (excluding the amount borrowed) in real estate and silver combined, and
- the average risk of the portfolio should not exceed 10.

In order to formulate the appropriate model, we first must define the problem variables. The scarce resource in this application is money, so that we define x_j as the dollar amount invested in the j-th alternative. Here, x_1 will denote the money invested in real estate, x_2 the money invested in silver, and so forth.

We can then formulate the objective function. Each dollar invested in real estate will produce no interest or dividend, but will gain an expected 18%, so that the investment of $1x_1$ will have appreciated to $1.18x_1$. A dollar invested in silver and a savings account will appreciate to \$1.10 and \$1.02, respectively. A \$1 investment in blue chip stocks is expected to be worth \$1.06 after a year plus a dividend of 3¢, making it worth \$1.09). The remaining investments are dealt with in a similar fashion. From our revenue we must deduct the amount borrowed (x_7) plus the interest to be paid on the borrowed amount ($0.12x_7$), making it $1.12x_7$.

Consider now the constraints. An obvious restriction is that the investor cannot invest more money than is available. The amount invested is nothing but the sum of all individual investments, i.e., $x_1 + x_2 + x_3 + x_4 + x_5 + x_6$, while the amount that is available is the sum of the \$300,000 at hand plus the amount that is borrowed, *viz*., x_7. This is the budget constraint (1). Another straightforward restriction limits the amount that can be borrowed to \$100,000. This constraint is shown in (2).

The constraint that requires the invested money to show a growth of at least 7% can now be formulated as follows. The actual value of the assets at the end of the planning period uses the expected annual gains from Table 2.2.6, resulting in $1.18x_1$, $1.10x_2$, etc. The required increase of the invested money by at least 7% is the product of 1.07 (the principal plus the required increase) and the invested amount, which is the sum of the first six variables. This is shown in Constraint (3).

The constraints (4) and (5) model the requirements that at least 50% of the money invested must be invested in stocks and bonds combined, and that no more than 20% of the money available can be invested in real estate and silver combined. Note the difference between the two constraints: while in (4), we are dealing with

a portion of the amount actually invested (the sum of the first six variables), constraint (5) refers to the total amount available to the investor (exclusive the amount that may be borrowed), which equals \$300,000.

The average risk of the portfolio equals the total risk divided by the amount invested, i.e., $\frac{20x_1 + 12x_2 + x_3 + 7x_4 + 3x_5 + 30x_6}{x_1 + x_2 + x_3 + x_4 + x_5 + x_6}$, which is not to exceed a value of 10. Multiplying the inequality by the (nonzero) denominator results in relation (6). The nonnegativity constraints (7) conclude the formulation. Version 1 of the model can then be summarized as follows.

$$\text{P: Max } z = 1.18x_1 + 1.10x_2 + 1.02x_3 + 1.09x_4 + 1.04x_5 + 1.20x_6 - 1.12x_7$$
$$\text{s.t. } x_1 + x_2 + x_3 + x_4 + x_5 + x_6 \le 300{,}000 + x_7 \quad (1)$$
$$x_7 \le 100{,}000 \quad (2)$$
$$1.18x_1 + 1.10x_2 + 1.00x_3 + 1.06x_4 + 1.00x_5 + 1.20x_6 \ge 1.07(x_1 + x_2 + x_3 + x_4 + x_5 + x_6) \quad (3)$$
$$x_4 + x_5 + x_6 \ge 0.5(x_1 + x_2 + x_3 + x_4 + x_5 + x_6) \quad (4)$$
$$x_1 + x_2 \le 0.2(300{,}000) \quad (5)$$
$$20x_1 + 12x_2 + x_3 + 7x_4 + 3x_5 + 30x_6 \le 10(x_1 + x_2 + x_3 + x_4 + x_5 + x_6) \quad (6)$$
$$x_1, x_2, \ldots, x_7 \ge 0. \quad (7)$$

Table 2.2.7 below shows the solution of this model with a 12% interest on borrowed money (as formulated above), and a slight modification with 10% interest on the amount we borrow.

Table 2.2.7: Optimal solutions to the different versions of the investment allocation model

Investment type	Version 1 with 12% interest on borrowed money	Version 1 with 10% interest on borrowed money	Version 2
Real estate	60,000	60,000	0
Silver	0	0	0
Savings account	0	0	160,500
Blue chip stocks	234,782.61	321,739.13	0
Bonds	0	0	160,500
Hi-tech stocks	5,217.39	18,260.87	0
Amount borrowed	0	100,000	21,000

Version 2 of the investment model is very similar. It deletes constraint (6) from the formulation and uses its left-hand side (the actual risk of the portfolio) in a minimization objective. Furthermore, we require an appreciation of at least 7% on

the available money (exclusive the borrowed amount), i.e., \$321,000 at the end of the planning period. A summary of the optimal solutions is shown in Table 2.2.7.

Note the jump that occurs in the amount of money borrowed: apparently, given an interest of 10% on borrowed money, it is still worthwhile to borrow and invest, while an increase to 12% is prohibitive, so that no more money is borrowed and no leveraging occurs. Also note the very different solution provided by Version 2 of the model.

Another allocation problem is found in the allocation of manpower—the scarce resource in this context—to tasks. In this specific application, we deal with allocating police officers to districts. Clearly, not all districts in a city are created equal, so that the impact a single policeman provides to a district will be different for different areas. In the numerical illustration below, we have a total of 67 police officers to distribute among five districts *A*, *B*, …, and *E*. Table 2.2.8 shows the degree of protection offered to a district for each officer assigned to the district.

Table 2.2.8: Protection provided by police officers and smallest allowable protection

	A	*B*	*C*	*D*	*E*
Additional marginal protection per officer	3	7	10	5	4
Lowest acceptable protection	40	50	70	60	40

For example, if six police officers are assigned to district *A*, then the degree of protection is 6(3) = 18. Suppose that due to the intricacies of the job, it is not possible to hire additional police officers in the short run. Actually, each police officer who is not assigned to one of the districts can be hired out to a private security company for a fee of \$100 per officer and day. Other than the house taxes (a flat fee for our purposes that does not have to be included in the problem), hiring out police officers is the council's only income. Council wants to set up a linear programming problem that maximizes the department's income (due to hiring out police officers). The following restrictions have to be observed:

- It must be ensured that each district has at least the minimum protection as specified in the table above,
- the average protection is at least 50, and
- we have to allocate at least 50% more officers to districts *A*, *B*, and *C* combined than to districts *D* and *E* combined.

Formulating the problem, we have to define the variables first. The scare resource in this context are police officers, so that we can define x_j as the number of police officers assigned to district *j*. For simplicity, we use x_A, x_B, x_C, x_D, and x_E for the five districts under consideration. The objective function then maximizes the product

of the revenue for a police offers hired out for a day ($100) and the number of police offers hired out. The number of police officers hired out for private security work equals the number of police officers available (67) and the number of officers assigned to the five districts (the sum of all five variables).

The first constraint simply requires that we cannot assign more police officers than we have. Constraints (2) – (6) ensure that each district receives at least the minimum protection required in Table 2.2.7. Constraint (7) requires that the average protection in the five distracts is at least 50, and constraint (8) ensures that the first three districts have at least twice as many police officers allocated to them as have the last two. As usual, the nonnegativity constraints complete the formulation. The model can then be written as follows.

$$
\begin{array}{lrcl}
\text{P: Max } z = 100[67 - (x_A + x_B + x_C + x_D + x_E)] & & & \\
\text{s.t.} \quad x_A + x_B + x_C + x_D + x_E & \le 67 & & (1) \\
3x_A & \ge 40 & & (2) \\
7x_B & \ge 50 & & (3) \\
10x_C & \ge 70 & & (4) \\
5x_D & \ge 60 & & (5) \\
4x_E & \ge 40 & & (6) \\
(3x_A + 7x_B + 10x_C + 5x_D + 4x_E)/5 & \ge 50 & & (7) \\
x_A + x_B + x_C & \ge 1.5(x_D + x_E) & & (8) \\
x_A, \; x_B, \; x_C, \; x_D, \; x_E & \ge 0. & & (9)
\end{array}
$$

Solving the problem reveals that the number of police officers allocated to the five districts are 13⅓, 12⅔, 7, 12, and 10, respectively (for now, we will disregard the nonintegralities). This means that a total of 55 police officers are allocated, leaving 12 police officers to be hired out, so that city council's daily income from this allocation is $1,200. Furthermore, this allocation results in protection levels of 40, 88⅔, 70, 60, and 40. In other words, all districts except the second receive the minimal protection required, while district *B* receives 88⅔/50 ≈ 1.77 times the protection that is minimally needed.

2.2.4 Employee Scheduling

The model in this section is, in some sense, also an allocation problem. However, it has its own character, so that it justifies its own section. Consider a recurring situation in which employees have to be assigned to shifts. The number of employees required to be on the job varies throughout the day during a variety of time slots. For instance, a bus route will require significant service during the early morning and afternoon rush hours, while there will not be much service during lunch hour or late at night. Similar requirements exist for nurses, pilots, cashiers in grocery stores, and similar scenarios.

The difficulty with this problem is that we are typically not able to hire casual labor whenever needed, but we will have to use permanent employees. So the

objective of the problem is to use the smallest number of employees and still be able to staff the position(s) throughout the day.

In our numerical example, assume that a regular shift is 8 hours and assume that there are 4-hour time segments during which personnel requirements have been observed. The personnel requirements during the 4-hour time slots are shown in Table 2.2.9 using a 24-hour clock.

Table 2.2.9: Personnel requirements during 4-hour time slots

Shift	0600–1000	1000–1400	1400–1800	1800–2200	2200–0200	0200–0600
Required number of employees	17	9	19	12	5	8

Assume that shift work can start every four hours at 6 a.m., 10 a.m., and so forth. Our decision is then how many employees to hire at each of these points in time. This means that we can define variables x_{06}, x_{10}, x_{14}, x_{18}, x_{22} and x_{02} as the number of employees who start their shift at 6 a.m., 10 a.m., 2 p.m., and so forth. The total number of employees required is then the sum of all of these variables. As far as the constraints go, we have to require that a sufficient number of employees is present during each time slot. Consider, for instance, the time slot between 1400 and 1800 hours, during which at least 19 employees are needed. The employees working during this time slot are those whose shift starts at 1000 hours plus those who start working at 1400 hours. This means that during this time slot $x_{10} + x_{14}$ employees will be working, a number that must be at least 19. Similar constraints have to be formulated for all six time slots. The formulation can then be written as follows, where we ignore the integrality requirements for reasons of simplicity.

$$
\begin{aligned}
\text{P: Min } z = {} & x_{06} + x_{10} + x_{14} + x_{18} + x_{22} + x_{02} \\
\text{s.t.} \quad & x_{06} + x_{02} \geq 17 \\
& x_{06} + x_{10} \geq 9 \\
& x_{10} + x_{14} \geq 19 \\
& x_{14} + x_{18} \geq 12 \\
& x_{18} + x_{22} \geq 5 \\
& x_{22} + x_{02} \geq 8 \\
& x_{06},\ x_{10},\ x_{14},\ x_{18},\ x_{22},\ x_{02} \geq 0.
\end{aligned}
$$

Problems of this type typically have multiple solutions. The problem as formulated has an optimal solution that requires a total of 41 employees. The starting times of their shifts are shown in Table 2.2.10.

Table 2.2.10: Starting times of employees in optimal solution for 4-hour time slots

Start of shift	0600	1000	1400	1800	2200	0200
Number of employees	14	7	12	0	5	3

Note that this solution has the exact number of required employees during all time slots, except for the time 1000-1400, where only 9 employees are needed, while 21 employees are available. In other words, there are 12 employees idle between 10 a.m. and 2 p.m.

As an extension of the above model, assume that it is now possible to start the employees' shifts each two hours rather than each four hours. Similarly, the time requirements are known for 2-hour rather than four-hour segments throughout the day. For instance, the 17 employees that were needed between 6 a.m. and 10 a.m. in the above problem, are required between 6 a.m. and 8 a.m., while between 8 a.m. and 10 a.m. only 11 employees are required. The personnel requirements during the 2-hour time slots are shown in Table 2.2.11. Note that the larger requirement of each two adjacent time slots that correspond to a 4-hour time slot in the above example equals the requirement of that 4-hour slot. In that sense, we are using the same example, just a finer grid.

Table 2.2.11: Personnel requirements during 2-hour time slots

Shift	0600–0800	0800–1000	1000–1200	1200–1400	1400-1600	1600–1800
Required number of employees	17	11	9	7	13	19

Shift	1800–2000	2000–2200	2200–2400	2400–0200	0200–0400	0400–0600
Required number of employees	12	8	5	3	3	8

The problem can then be formulated as follows.

$$
\begin{aligned}
\text{P: Min } z = {} & x_{06} + x_{08} + x_{10} + x_{12} + x_{14} + x_{16} + x_{18} + x_{20} + x_{22} + x_{24} + x_{02} + x_{04} \\
\text{s.t.} \quad & x_{06} + x_{24} + x_{02} + x_{04} \geq 17 \\
& x_{06} + x_{08} + x_{02} + x_{04} \geq 11 \\
& x_{06} + x_{08} + x_{10} + x_{04} \geq 9 \\
& x_{06} + x_{08} + x_{10} + x_{12} \geq 7 \\
& x_{08} + x_{10} + x_{12} + x_{14} \geq 13 \\
& x_{10} + x_{12} + x_{14} + x_{16} \geq 19 \\
& x_{12} + x_{14} + x_{16} + x_{18} \geq 12 \\
& x_{14} + x_{16} + x_{18} + x_{20} \geq 8 \\
& x_{16} + x_{18} + x_{20} + x_{22} \geq 5 \\
& x_{18} + x_{20} + x_{22} + x_{24} \geq 3 \\
& x_{20} + x_{22} + x_{24} + x_{02} \geq 3 \\
& x_{22} + x_{24} + x_{02} + x_{04} \geq 8 \\
& x_{06}, \; x_{08} \; x_{10}, \; x_{12}, \; x_{14}, \; x_{16}, \; x_{18}, \; x_{20}, \; x_{22}, \; x_{24}, \; x_{02}, \; x_{04} \geq 0.
\end{aligned}
$$

The optimal solution of this problem requires now only 36 employees, and the starting times are shown in Table 2.2.12.

Table 2.2.12: Starting times of employees in optimal solution for 2-hour time slots

Start of shift	06	08	10	12	14	16	18	20	22	24	02	04
Number of employees	0	0	7	4	3	5	0	0	0	3	0	14

Particularly noteworthy are the more than 12% savings in the number of employees that must be hired. In general, it is not surprising that the finer grid used here provides a solution that is at least as good as that with 4-hour time slots. The reason is that the previous solution can still be implemented and it would still provide a feasible solution. However, with the additional possible starting times there are additional possibilities which may—and in this case do—allow us to find a better solution.

2.2.5 Dynamic Production – Inventory Models

This section describes models, in which decision makers do not only have to answer the "how many" question as we have seen in many of the previous applications, but they also require an answer to the question "when" to produce. In the simplest case, assume we only consider a single product. Furthermore, suppose that the time frame of interest has been subdivided into small time units, in which production occurs. Throughout this section, we will refer to these units as "months." Within each month, production occurs and customers take out products based on their demand.

Based on the production capacities, it may now not be possible to satisfy the demand in each month. In order to avoid undersupplying our customers, we can produce more than the demand indicates during the earlier months of the planning period and keep the surplus in stock. This will, of course, cause inventory holding costs to be incurred. Given that the production costs may vary between the months, it may actually be preferable to manufacture goods earlier in the planning period rather than later, but the decision will depend on the relation between the production costs and the inventory holding costs. We will leave these decisions to the optimizer and our model.

Before presenting a numerical example, it is important to discuss the exact sequence of events within each month. At the beginning of each month, we take stock. Since nothing has happened to the inventory between this point in time and the previous month, the inventory level at the beginning of the month will denote the number of units carried over from the previous month, which, in turn, will determine the inventory holding costs. Then production occurs. After the desired production quantity is made, customers take products out of our warehouse according to the estimated demand. Whatever is left after that will be carried over to the next month, and the process begins anew.

As a numerical illustration, consider the following scenario. The planning period ranges from the beginning of January of some year and ends at the end of April.

The estimated demand, production capacity, and unit production costs are shown in Table 2.2.13.

Table 2.2.13: Parameters for the dynamic production – inventory model

	Month 1 (January)	Month 2 (February)	Month 3 (March)	Month 4 (April)
Estimated demand	80	70	130	150
Production capacity	120	140	150	140
Unit production cost	\$1.00	\$1.10	\$1.20	\$1.25

In addition, it costs 5¢ to carry over one unit from the end of January to the beginning of February, 15¢ to carry over one unit from the end of February to the beginning of March, and another 15¢ to hold one unit in stock between the end of March and the beginning of April. The decision maker has an opening inventory of 20 units in the beginning of the planning period and desires to have nothing left at the end of the planning period.

In order to formulate the model, we quite naturally need two types of variables, one for production and the other for inventory. Denote the production variables by x_1, x_2, x_3, and x_4, which are defined as the quantities to be manufactured in months 1, 2, 3, and 4, respectively. Similarly, we define the parameters d_1, d_2, d_3, and d_4 as the demand in periods 1, 2, 3, and 4, respectively. Before defining the inventory variables, we have to decide at which point to measure the inventory level. Given our problem description above, we may decide to count inventory at the beginning of each period. Alternatively, it is possible to determine the inventory level at the end of a period, which we leave as exercise in Problem 3 at the end of this section. Here, the we denote by I_1, I_2, I_3, I_4, and I_5 the inventory levels at the beginning of the periods 1 to 5. Note that the inventory levels I_1 and I_5 are not variables, but parameters whose numbers we know: as outlined above, the opening inventory $I_1 = 20$, and the closing inventory $I_5 = 0$.

The objective function is then a simple cost minimization function that consists of two main components, the production costs and the inventory costs. As far as constraints go, there are two types. First, there are the simple production capacity constraints that specify that in no period can we produce more than our capacity allows. Secondly, there are the inventory balancing constraints. They state that the inventory level at the beginning of period t equals the inventory level at the beginning of the previous period $t-1$ plus our production in the previous period minus the demand in the previous period. Formally, we can write $I_t = I_{t-1} + x_{t-1} - d_{t-1}$ for $t = 2$ to $n+1$, where n is the last month within the time frame. These constraints are the same as the usual balancing constraints in accounting that state that what you have in your account today equals what you had yesterday plus the deposits yesterday minus yesterday's withdrawals. Our model can then be formulated as follows.

$$\text{Min } z = 1x_1 + 1.1x_2 + 1.2x_3 + 1.25x_4 + .05I_2 + .15I_3 + .15I_4$$

$$\begin{aligned}
\text{s.t. } x_1 &\leq 120 \\
x_2 &\leq 140 \\
x_3 &\leq 150 \\
x_4 &\leq 140 \\
I_2 &= I_1 + x_1 - 80 \text{ (or, as } I_1 = 20,\ x_1 - I_2 = 60) \\
-I_3 + I_2 + x_2 &= 70 \\
-I_4 + I_3 + x_3 &= 130 \\
-I_5 + I_4 + x_4 &= 150 \text{ (or, as } I_5 = 0,\ x_4 + I_4 = 150) \\
x_1, x_2, x_3, x_4, I_2, I_3, I_4 &\geq 0.
\end{aligned}$$

The optimal production schedule has us manufacture 120, 10, 140 and 140 units of the product in the four months, and the inventories carried over between months 1 and 2, 2 and 3, and 3 and 4 are 60, 0, and 10, respectively. The sum of the production and inventory costs is \$478.50.

The solution makes intuitive sense, as the low production level in February is a result of the lower production costs in January and the low inventory costs between January and February. On the other hand, the inventory carrying costs are significant after February, so that inventories only occur between March and April, and these are necessary as the April demand exceeds the production capacity in that month.

A potential extension of the model may consider warehouse capacities. In other words, we may impose limits on the number of units we can keep in stock. Such constraints are easily incorporated in this formulation. If in our numerical example the largest possible inventory levels between months 1 and 2, months 2 and 3, and months 3 and 4 are 40, 50, and 50, respectively, we add the constraints

$$\begin{aligned}
I_2 &\leq 40 \\
I_3 &\leq 50 \\
I_4 &\leq 50.
\end{aligned}$$

With these additional constraints, the production levels in the four periods are revised to 100, 30, 140, and 140, respectively, so that the inventory levels between the periods are 40, 0 and 10, respectively. The total costs for this system then (marginally) increase to \$479.50.

This problem can also be formulated in an alternative fashion. Rather than using separate variables for production and inventory, we can define double-subscripted variables x_{ij} that indicate how many units of the product were manufactured in month i for use in month j. Such a formulation will require some additional preprocessing. For instance, in order to determine the objective function coefficient for the variable x_{14} in the numerical example in this section, we need to add the production costs in month 1 (when the product is made) and the inventory holding costs from month 1 to 2, those from month 2 to 3, and those from month 3 to 4 for

a total of 1 + .05 + .15 + .15 = \$1.35. The other coefficients in the objective function are determined similarly.

In addition, there will be two sets of constraints. The first are again the constraints that require the production capacities to be respected. For instance, the total production in Month 1 will be $x_{11} + x_{12} + x_{13} + x_{14}$, the production in Month 2 will be $x_{22} + x_{23} + x_{24}$, and similar for the remaining two months. The second set of constraints that are needed are then the demand constraints. The number of units available in, say, Month 3 is $x_{13} + x_{23} + x_{33}$ and this number must be at least as large as the demand in that month. The problem can then be formulated as follows.

$$\text{P: Min } z = 1x_{11} + 1.05x_{12} + 1.2x_{13} + 1.35x_{14} + 1.1x_{22} + 1.25x_{23} + 1.4x_{24} + 1.2x_{33} + 1.35x_{34} + 1.25x_{44}$$

$$\text{s.t. } x_{11} + x_{12} + x_{13} + x_{14} \leq 120$$
$$x_{22} + x_{23} + x_{24} \leq 140$$
$$x_{33} + x_{34} \leq 150$$
$$x_{44} \leq 140$$
$$x_{11} \geq 60 \text{ (January's is reduced by the available opening inventory of 20 units)}$$
$$x_{12} + x_{22} \geq 70$$
$$x_{13} + x_{23} + x_{33} \geq 130$$
$$x_{14} + x_{24} + x_{34} + x_{44} \geq 150$$
$$x_{11}, x_{12}, x_{13}, x_{14}, x_{22}, x_{23}, x_{24}, x_{33}, x_{34}, x_{44} \geq 0.$$

The optimal solution can best be summarized in a table such as that shown in Table 2.2.14.

Table 2.2.14: Optimal solution of the production-inventory problem

	Month 1	Month 2	Month 3	Month 4
Month 1	60	60	0	0
Month 2	–	10	0	0
Month 3	–	–	130	10
Month 4	–	–	–	140

The cost at optimum are \$478.50, obviously the same as in the other formulation. The actual production levels in the four months can be determined by adding the values of the variables in the rows, while the sums in the columns result in the months' demand. It is also a good idea to plot the inventory changes on a time line that lists the opening inventory of each month, adds the production within the month, and then subtracts the demand later that month. This is shown in Figure 2.2.1.

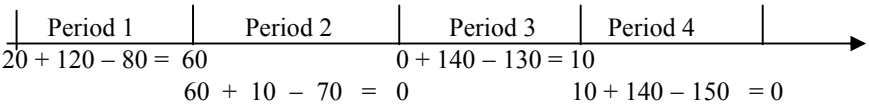


Figure 2.2.1

Hence, the inventory levels at the beginning of the periods 2, 3, and 4 are 60, 0, and 10, respectively; again, reflecting the same result obtained earlier.

Incorporating limits on the warehouse capacity is also easy in this formulation. The number of units put in stock between January and February is $x_{12} + x_{13} + x_{14}$, the level of stock between February and March is $x_{23} + x_{24}$, and the level between March and April is x_{34}. All that needs to be done is to require these expressions not to exceed 40, 50, and 50, respectively, and we will again obtain the same result computed earlier for the formulation with explicit inventory variables.

Which of the formulations is used depends on the preferences of the user. The former model with the explicit inventory variables has the advantage of having $2n$ variables, given again n months within the planning period, while the latter model with the double-subscripted variables requires $\frac{1}{2}n^2$ variables. However, since the value of n is typically quite small and modern linear programming solvers can easily deal with formulations that have hundreds of thousands of variables, this should not be a problem.

2.2.6 Blending Problems

All blending problems have in common that they take a number of given ingredients or raw materials and blend them in certain proportions to the final products. In other words, in the process we are creating something new by mixing existing materials. Typical examples for blending are coffees, teas, whiskeys, tobaccos, and similar products. Clearly, there are some rules for the blending process. For instance, in order to ensure a specific taste, it may be required that a blend includes at least a certain proportion of a raw material.

The process is shown in Figure 2.2.2. On the left, there are m buckets with given quantities of raw materials, while on the right, there are n empty buckets that are to be filled with known quantities of the blends. In the blending process we take, one at a time, a certain number of scoops from each of the raw materials and transfer them into the buckets on the right. Once sufficient quantities have been transferred to the "Product" buckets on the right, all that is left to do is stir, package, and sell.

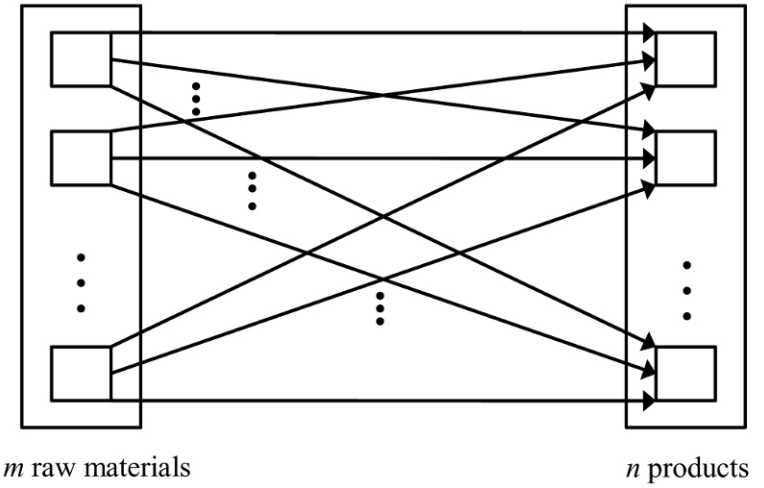


Figure 2.2.2

This figure not only demonstrates the actual process, but it also allows us to see how the variables in blending problems are to be defined. What we need to know in any specific blending problem is how many "scoops" of each raw material goes into each of the "Product" buckets. In more general terms, we define x_{ij} as the quantity of raw material i that goes into product j.

As a numerical example, suppose that we are to blend two table wines from the Moselle region in Germany. The two blends are the *Filzener Hexenhammer* and the *Leiwener Hosenscheisser*. Both products are blends of wines from three grapes, *viz*., *Riesling*, *Müller-Thurgau*, and *Silvaner*. The original wines are available in quantities of 10,000, 5,000, and 6,000 gallons at a cost of \$8, \$6, and \$5 per gallon. The estimated demands for the two blends are 7,000 and 8,000 gallons and the estimated sales prices are \$16 and \$18 per gallon.

The rules that have to be followed when blending the two wines are summarized in Table 2.2.15. The meaning of these figures is best explaind by some of the numbers. For instance, the interval [.45; .55] in the *Riesling* row and the *Filzener Hexenhammer* column of the table indicates that at least 45% and at most 55% of the *Filzener Hexenhammer* blend must be *Riesling*. Note that the *Silvaner* content of the *Hexenhammer* must be exactly 35%. It is also noteworthy that while the ranges for the *Hexenhammer* blend are quite tight (which usually indicates a well-controlled and high-quality product), the ranges for the *Hosenscheisser* are very wide, clearly indicating a cheap product that consists of, loosely speaking, more or less whatever happens to be available.

Table 2.2.15: Blending rules for the wines

Blends / Basic wines	*Filzener Hexenhammer*	*Leiwener Hosenscheisser*
Riesling	[.45; .55]	[.20; .50]
Müller-Thurgau	[.10; .15]	[.10; .60]
Silvaner	[.35; .35]	[.30; .40]

As discussed above, we need six variables in this example. They are x_{11} (the quantity of *Riesling* in the *Hexenhammer*), x_{12} (the quantity of *Riesling* in the *Hosenscheisser*), x_{21} (the quantity of *Müller-Thurgau* in the *Hexenhammer*), and x_{22}, x_{31}, and x_{32} which are defined analogously. Before formulating the objective, it is beneficial to first determine the quantities of the basic wines that are used in the blending process and the quantities of the blends that are made in the process. For the time being, let us assume that there are no losses in the process (e.g., spillage or thirsty employees).

Consider the basic wines and their uses first. Each of those wines is either put into the *Hexenhammer* or the *Hosenscheisser* blend—that is all we can do with them in this context. Thus the quantity of *Riesling* that we use is $x_{11} + x_{12}$, the quantity of *Müller-Thurgau* used in the process is $x_{21} + x_{22}$, and the quantity of *Silvaner* that is used is $x_{31} + x_{32}$. Similarly we determine the quantities of the blends that are produced. The quantity of each blend that is made in the process is nothing but the sum of its ingredients. In our example, the quantity of *Hexenhammer* that we blend equals $x_{11} + x_{21} + x_{31}$, and the quantity of *Hosenscheisser* we make is $x_{12} + x_{22} + x_{32}$. The objective function is then to maximize profit, which, in turn equals the difference between revenues from the two blends and the costs of the three basic wines. It is shown as relation (1) below.

As far as constraints go, we have three different types of constraints. First there are the supply constraints that require not to use more of the basic wines than we can get, secondly there are the demand constraints that state that we have to make at least as many gallons of the blends as our customers are estimated to demand, and thirdly and finally, there are the blending constraints.

Since the quantities of the ingredients (the basic wines) and the products (the blends) have already been determined above, the first two types of constraints are easy to formulate. They are shown as constraints (2) and (3) in the formulation below. The blending constraints (4) are formulated as follows. Consider the content of *Riesling* in *Hexenhammer*. One of the blending constraints will state that the quantity of *Riesling* in *Hexenhammer* (which is x_{11}) must be at least 45% of the total quantity of *Hexenhammer*, which has already been determined as $x_{11} + x_{21} + x_{31}$. In other words, we can write $x_{11} \geq .45(x_{11} + x_{21} + x_{31})$. This is the lower bound of this combination of basic wine and blend. The upper bound is formulated in similar fashion as $x_{11} \leq .55(x_{11} + x_{21} + x_{31})$. This is repeated for all combinations of basic wines and blends. The complete formulation is then

$$\text{Max } z = [16(x_{11} + x_{21} + x_{31}) + 18(x_{12} + x_{22} + x_{32})] - [8(x_{11} + x_{12}) + 6(x_{21} + x_{22}) + 5(x_{31} + x_{32})] \quad (1)$$

$$\text{s.t.} \quad \begin{aligned} x_{11} + x_{12} &\le 10{,}000 \\ x_{21} + x_{22} &\le 5{,}000 \\ x_{31} + x_{32} &\le 6{,}000 \end{aligned} \quad (2)$$

$$\begin{aligned} x_{11} + x_{21} + x_{31} &\ge 7{,}000 \\ x_{12} + x_{22} + x_{32} &\ge 8{,}000 \end{aligned} \quad (3)$$

$$\begin{aligned} x_{11} &\ge .45(x_{11} + x_{21} + x_{31}) \\ x_{21} &\ge .1(x_{11} + x_{21} + x_{31}) \\ x_{31} &\ge .35(x_{11} + x_{21} + x_{31}) \\ \\ x_{12} &\ge .2(x_{12} + x_{22} + x_{32}) \\ x_{22} &\ge .1(x_{12} + x_{22} + x_{32}) \\ x_{32} &\ge .3(x_{12} + x_{22} + x_{32}) \\ \\ x_{11} &\le .55(x_{11} + x_{21} + x_{31}) \\ x_{21} &\le .15(x_{11} + x_{21} + x_{31}) \\ x_{31} &\le .35(x_{11} + x_{21} + x_{31}) \\ \\ x_{12} &\le .5(x_{12} + x_{22} + x_{32}) \\ x_{22} &\le .6(x_{12} + x_{22} + x_{32}) \\ x_{32} &\le .4(x_{12} + x_{22} + x_{32}) \end{aligned} \quad (4)$$

$$x_{11}, x_{12}, x_{21}, x_{22}, x_{31}, x_{32} \ge 0.$$

The optimal solution of the problem uses 3,850 gallons of *Riesling*, 700 gallons of *Müller-Thurgau*, and 2,450 gallons of *Silvaner* in the *Hexenhammer*, so that a total of 7,000 gallons of that blend are made. Note that this is exactly the quantity that is desired. Similarly, we use 3,983.33 gallons of *Riesling*, 4,300 gallons of *Müller-Thurgau*, and 3,550 gallons of *Silvaner* in the *Hosenscheisser* blend, making it a total of 11,833.33 gallons of that blend. This is considerably more than the minimum of 8,000 gallons that were required. Also note that in the process we use a total of 7,833.33 gallons of *Riesling* (with 2,166.67 gallons left over), we use all of the 5,000 gallons of *Müller-Thurgau* and all of the 6,000 gallons of *Silvaner* that are available to us. The overall profit is $202,333.33.

It is easy to add other relevant constraints. Suppose that we need to control the alcoholic content of the blends as well. For that purpose, assume that the *Riesling* has an alcohol content of 8%, *Müller-Thurgau* has 7.5%, and *Silvaner*

has 6%, and it is desired that the *Hexenhammer* has at least 7.3% alcohol content. Note that the present solution contains .08(3,850) + .075(700) + .06(2,450) = 507.5 gallons of alcohol. Given that we are making 7,000 gallons of the blend, this means that the present alcohol content of *Hexenhammer* is 507.5/7,000 = 7.25%, which is not sufficient. The constraint requiring at least 7.3% can be written as $.08x_{11} + .075x_{21} + .06x_{31} \geq .073(x_{11} + x_{21} + x_{31})$. The new optimal solution does not change the quantity or composition of *Hosenscheisser*, but dramatically changes the composition of *Hexenhammer*. It now contains 5,916.67 gallons *Riesling*, 2,366.67 gallons of *Müller-Thurgau*, and 3,350 gallons of *Silvaner*. The profit has decreased to $198,466.7, roughly a 2% decrease.

2.2.7 Transportation and Assignment Problems

Transportation problems have been introduced into the discussion by Hitchcock in 1941, thus predating the advent of linear programming by half a dozen years. It must be understood that there is not a single model that encompasses all, or even most, transportation scenarios. What we call "the" transportation problem in this context is a very simple prototype that exhibits some basic structures that are typically inherent in transportation problems. The structure has three essential components. On the one hand, there are *origins*, where certain quantities of a single homogeneous good are available (so that one unit of the good is exactly the same, regardless from which origin it is taken). We can think of the origins as warehouses at which goods are stored. For simplicity, we will use the terms warehouse and origin interchangeably. Given that there is a total of m origins, we assume that there is a known supply of s_i at origin i.

The second set of components are the *destinations*. This is where certain quantities of the good are needed. We will refer to them as either destinations or customers. We assume that there are n destinations, and suppose that there is a known demand for our product at a magnitude of d_j at destination j. So far, we have existing goods on one side, and the need for goods (or, equivalently, existing demand, on the other. The task is now to ship the existing goods from the origins to the destinations, so as to respect the existing quantities at the origins, satisfy the demand at the destinations, and organize the shipments as efficiently as possible. As a matter of fact, we can visualize the structure of the problem by considering the graph used in our discussion of blending problems (Figure 2.2.2). Rather than input factors on the left and blended products on the right, we have origins and destinations, and instead of taking a number of ladles full of raw materials and put them into a bucket for a blended product, we ship a number of units from a warehouse to a customer. The visuals and the story are very different, but the mathematical structure is the same.

When requiring efficiency of the shipments, we will need a criterion by which to measure efficiency. As usual, we will employ a monetary criterion. Here, an obvious choice are the costs of transportation. In order to introduce cost components, we assume that there is a cost of c_{ij} to ship a single unit of the good from origin i to

destination j. Fixed unit cost of transportation mean a linear transportation cost function, as one unit shipped from i to j costs c_{ij}, two units costs $2c_{ij}$, and so forth. The assumption of linearity of the cost function is crucial to our model. Assume that we are shipping pallets of bottles from regional distribution centers to supermarkets. Suppose now that the capacity of a truck is forty pallets. Then the transportation cost for one pallet is one trip, the costs for two pallets are one trip (except for the cost of loading and unloading the same as the cost for a single pallet), the costs for three pallets are still mainly the costs of a single trip, etc. In other words, its costs about the same to ship one pallet or up to forty pallets. Shipping pallet number 41, though, requires a second trip, causing the costs to jump to the equivalent of two trips. The result is a step function with break points at the truck capacities. One way to "restore" the linearity assumption in this model is to measure the number of units shipped in terms of truckloads. Disregarding the differences in loading and unloading costs for different numbers of pallets, the cost function is now linear. It is very important to note that the transportation network in this basic problem allows only direct transportation from an origin to a destination. It is *not* permitted to first ship from one origin to another to consolidate the load, to use round trips (similar to multi-stop shopping), or similar non-direct routes.

Before formulating a small example, it is necessary to distinguish between balanced and unbalanced transportation problems. A transportation is called *balanced*, if the sum of all supplies equals the sum of all demands. In balanced transportation problems, it will be possible to empty all the warehouses and satisfy all demands of our customers. In *unbalanced* transportation problems, we either have more units than customers want (the total supply exceeds the total demand), or we need more units than we have (the total demand exceeds the total supply). In the former case we will be able to satisfy all demand, but some units will be left over, while in the latter case, we will empty all the warehouses, but some demand will remain unsatisfied. We will discuss the balanced case first, and then offer modifications that deal with unbalanced problems.

Consider a transportation problem with two origins and three destinations. The supplies at the origins are 30 and 20 units, respectively, while at the destinations, there are demands of 15, 25, and 10, respectively. Clearly, the total supply and the total demand both equal 50, making the problem balanced. The unit transportation costs are shown in the matrix **C** below, where an element in row i and column j shows the value c_{ij}. For instance, shipping one unit from origin 1 to destination 3 costs \$4.

$$\mathbf{C} = \begin{bmatrix} 1 & 7 & 4 \\ 2 & 3 & 5 \end{bmatrix}$$

The three types of parameters—the supplies, the demands, and the unit transportation costs—completely describe the transportation problem. The model will minimize the total transportation costs, while ensuring that supplies and demands are respected.

In order to formulate the problem, we define decision variables x_{ij} that denote the quantity that is shipped (directly) from origin i to destination j. The objective function is then composed as follows. First of all, we compute the total transportation costs along each origin – destination connection. As an example, in our numerical illustration the link between origin 2 and destination 1 carries unit transportation costs of \$2 and the quantity we ship on that link is x_{21}, making the total transportation cost on that connection $2x_{21}$. Adding such expressions on all links results in the overall total cost, which are then to be minimized in the objective function of the model as shown below.

Next, consider the constraints of the problem. Generally speaking, there will be two sets of constraints: the first set ensures (for balanced problems such as this example) that the flow of goods out of each origin equals exactly the quantity that is available at that origin. The second set of constraints requires that the flow of goods that is received at a customer site equals the demand of that customer. As an example for the first set of constraints, consider the second origin. From this origin, goods can be shipped (directly) to destination 1 (the quantity shipped on this link is x_{21}), to destination 2 (the quantity is x_{22}), and to destination 3 (with a quantity of x_{23}). Consequently, the total quantity shipped out of origin 2 is $x_{21} + x_{22} + x_{23}$ which is supposed to equal 20 units, the quantity available at origin 2. The remaining supply constraints are formulated similarly.

We are now able to formulate the demand constraints. As a numerical example, consider customer (or destination) 3. The goods that the customer receives either come from origin 1 (a quantity of x_{13}) and from origin 2 (a quantity of x_{23}). As a result the total quantity received at destination 3 is $x_{13} + x_{23}$, which should equal the demand of 10 at that point. Again, the other demand constraints are formulated in similar fashion. The complete formulation is then as follows:

$$\text{P: Min } z = 1x_{11} + 7x_{12} + 4x_{13} + 2x_{21} + 3x_{22} + 5x_{23}$$

$$\begin{aligned}
\text{s.t.} \quad & x_{11} + x_{12} + x_{13} = 30 \\
& x_{21} + x_{22} + x_{23} = 20 \\
& x_{11} + x_{21} = 15 \\
& x_{12} + x_{22} = 25 \\
& x_{13} + x_{23} = 10 \\
& x_{11}, x_{12}, x_{13}, x_{21}, x_{22}, x_{23} \geq 0.
\end{aligned}$$

The problem above is written in a way that makes the underlying structure clearly visible. It is apparent that each variable appears exactly twice in the constraints, once in a supply constraint and once in a demand constraint. The structure is very special and it ensures that as long as all supplies and demands are integer, there will exist at least one optimal solution to the problem that is also integer. The special structure of the problem has given rise to specialized solution techniques,

such as the *MODI* (*MO*dified *DI*stribution) method. With increasing computational power, their importance has diminished, so that we have chosen not to include them in this book, but instead include them on the website associated with this book. Another feature of the problem should be mentioned, even though we will not exploit it here. Given m origins and n destinations, the problem will have mn variables and $(m + n)$ constraints. However, one constraint is redundant (more specifically: linearly dependent). This can be seen in the above formulation by adding all supply constraints and then subtracting the first two demand constraints. The result will be the third demand constraint. As a result, we will have $(m+n-1)$ constraints to be considered. This number features prominently in the aforementioned special solution methods.

The optimal solution to our example can be shown in the following transportation plan **T**. In row i and column j, it shows the optimal value of the variable x_{ij}.

$$\mathbf{T} = \begin{bmatrix} 15 & 5 & 10 \\ 0 & 20 & 0 \end{bmatrix}.$$

In other words, we ship 15 units from origin 1 to destination 1, 5 units from origin 1 to destination 2, 10 units from origin 1 to destination 3, and all 20 units that are available at origin 2 directly to destination 2. It is easy to ascertain (by multiplying the elements of **T** with the corresponding elements of **C** and adding them up) that the total transportation cost at optimum is $\bar{z} = 150$. It is also worth mentioning that each nondegenerate solution to a transportation problem has exactly $(m+n-1)$ variables at a strictly positive level. In case of degeneracy, there may be fewer positive variables.

As far as extensions go, we will first look into unbalanced problems. First of all, given any unbalanced problem, we can no longer formulate the problem with all equations, as total supply and demand are no longer equal. In case of the total supply exceeding the total demand, we will not distribute the entire supply to our customers, so that some supply will be left over. This means that we can formulate the demand constraints as equations as we have in a balanced problem, while we write the supply constraints as less-or-equal-than constraints. That way, the customer demand is satisfied everywhere, while some units are left over in one or more of the origins. In the above example, let the supplies be 30 and 23 rather than 30 and 20. The optimal solution ships 27 units out of origin 1 and 23 out of origin 2, leaving three units unassigned at origin 1.

The case, in which the total supply falls short of the total demand is dealt with similarly. Here, we will not be able to satisfy the entire demand, but in order to come as close as possible of doing so, we will use our entire supply. This means that we formulate the supply constraints as equations, while the demand constraints will be written as less-than-or-equal constraints. In that way, the total supply is shipped to customers, which will still leave some unsatisfied demand at one or

more customers. In the above original example, suppose that the demands are now 15, 25, and 12, respectively, rather than 15, 25, and 10. The optimal solution ships 15 units to destination 1, 23 units to destination 2, and 12 units to destination 3, so that customer 2 will be left with an unsatisfied demand of 2 units. Extensions of the basic model may include penalties for unsatisfied customers (loss of goodwill) and units left over in a warehouse (inventory costs).

Two interesting extensions are called *reshipments* and *overshipments*. Both modifications are some type of sensitivity analysis, in that we change some of the existing assumptions. In the case of reshipments, we use the same transportation network, but allow transportation on routes that are not direct. In other words, a reshipment would be given if we were not to send a unit from, say, origin 1 to destination 1 directly, but ship it from origin 1 to destination 3 (or some other destination) first, then back to, say, origin 2, and from there on to destination 1. Clearly, in order to use reshipments (or back-and-forth shipments), it must be advantageous to do so. Consider again our example above. At optimum, we ship five units from origin 1 to destination 2. To ship a single unit on that route, it costs \$7. Instead, we could ship up to five units from origin 1 to destination 1 for \$1, back to origin 2 for an additional \$2, and from there to destination 2 for an additional \$3. Hence, it costs \$6 to ship a single unit on this somewhat circuitous route, \$1 less than on the direct connection. Reshipping five units this way will save 5(1) = \$5 for a total transportation costs of \$145. While reshipping may use routes that go back and forth multiple times, it is unlikely that such routes will exist in practice. The mathematical formulations of reshipments uses absolute values of variables, as the value of a variable such as $x_{ij} = -5$ indicates that five units are shipped back from destination j to origin i. This is not really problematic, but it makes the formulation somewhat more unwieldy.

Overshipments are another way of improving on the optimal solution to the basic problem by changing the assumptions somewhat. The idea is to allow additional flow through the transportation network, which, paradoxically, may actually reduce costs. Again, in our example consider the possibility to add one unit of supply to origin 2 (for a total of 21), and one unit of demand to destination 1 (for a total of 16). The optimal solution to that problem will move 51, rather than 50, units through the transportation network at a cost of 147, a decrease of \$3 from the original solution. Such a decrease is possible in this example, because we now ship one additional unit on the link from origin 1 to destination 1 (costing an additional \$1), and on the connection from origin 2 to destination 2 (costing an extra \$3), but we can now transport one less unit on the expensive link from origin 1 to destination 2 (which saves \$7). This explains the net savings of +1 + 3 – 7 = –\$3. While reshipments were easy to implement (just modify the plan of what is shipped where), overshipments are considerably more difficult to apply. In order to benefit from overshipments, we need additional units at the right origin, and we have to convince at least one customer to accept more units than originally demanded.

Other extensions of the basic problem have been discussed as well. One such extension includes capacities applied to the transportation links. Another, quite natural, modification includes not only direct shipments as used here, but allows transshipment points, at which the goods may be unloaded, temporarily stored, and reloaded onto other trucks. In some practical applications, such transshipment points do not just allow the consolidation of the loads, but also permit changing transportation modes, e.g., from truck to rail. Capacity constraints on the transportation points are a further natural feature to be included in a model. While some of these extensions may be incorporated in network models (see Chapter 5 of this volume), planners will generally resort to standard linear programming formulations to include all the desired features in the model.

A variety of other applications of "transportation problems" exist, some having absolutely nothing to do with shipping units from one place to another. One such example are the dynamic production – inventory models in Section 2.2.5. Here, the "origins" represent the periods of productions, while the "destinations" are the periods of consumption (or demand). A link from origin i to destination j exists, if $i \leq j$. The constraints then require that the outflows of the origins do not exceed the production capacities, while the inflows of the destinations must be at least as large as the known demand. Other applications assign groups of workers to shifts, or differently equipped military units to potential targets.

While transportation problems have a specialized structure, a*ssignment problems* are even more specialized. Consider a set of n employees that are to be assigned to n tasks. We can use no more than 100% of an employee time in the allocation, and to each task we must assign 100% of one or more employees' time. Each allocation bears a certain cost. As an example consider typists who, quite naturally, have different abilities. Suppose that one of the tasks involves technical typing. In order to perform the task, some typists may already know how to do it and can perform the type of task quite efficiently (meaning that there will be low costs of assigning this employee to the task), while other typists may have to be retrained, necessitating higher assignment costs. The problem is then to assign employees' time to tasks, so as to minimize the overall assignment costs.

In order to formulate the problem, we again first define the appropriate variables. Here, we define x_{ij} as the percentage of worker i's time that is assigned to task j. In order to explain the way the model is formulated, suppose that we have three employees and three tasks. The assignment costs are shown in the cost matrix

$$\mathbf{C} = \begin{bmatrix} 4 & 3 & 1 \\ 8 & 5 & 3 \\ 2 & 6 & 2 \end{bmatrix}.$$

If, for instance, we will use 20% of employee 1's time for task 2, then the costs are .2(3) = .6. The objective function will then minimize the sum of all assignment costs.

As far as the constraints of the model go, we have one set of constraints that specify that the sum of proportions of an employee's time must add up to 100%. Similarly, the sum of proportions of employees' time devoted to any one task must also add up to 100%. Given this structure, the problem can then be formulated as follows.

$$\text{P: Min } z = 4x_{11} + 3x_{12} + 1x_{13} + 8x_{21} + 5x_{22} + 3x_{23} + 2x_{31} + 6x_{32} + 2x_{33}$$

$$\begin{array}{lccccccccccc}
\text{s.t.} & x_{11} + & x_{12} + & x_{13} & & & & & & & & = 1 \\
 & & & & x_{21} + & x_{22} + & x_{23} & & & & & = 1 \\
 & & & & & & & x_{31} + & x_{32} + & x_{33} & & = 1 \\
 & & & & & & & & & & & \\
 & x_{11} + & & & x_{21} + & & & x_{31} & & & & & = 1 \\
 & & x_{12} + & & & x_{22} + & & & x_{32} & & & = 1 \\
 & & & x_{13} + & & & x_{23} + & & & x_{33} & & = 1 \\
 & & & & & & & & & & & \\
 & x_{11}, & x_{12}, & x_{13}, & x_{21}, & x_{22}, & x_{23}, & x_{31}, & x_{32}, & x_{33} & & \geq 0.
\end{array}$$

It becomes apparent that the assignment is a very close relative of the transportation problem discussed above. More specifically, since the constraints and the objective are exactly the same, we can view assignment problems as transportation problems with all supplies and demands equal to one. Given that, there will be at least one optimal solution to the problem that has all variables integer. It follows that at least one optimal solution has all variables either equal to zero or equal to one. In many publications, the variables are assumed to be one from the start, but this feature is really not an assumption of the general problem, but a consequence of the structure of the problem. It does, however, provide for an easier statement of the problem: the variables indicate whether or not an employee is assigned to a task (they equal one if he is and zero if not), and the constraints state that each employee is assigned to exactly one task, and each task is performed by exactly one employee.

Similar to the transportation problem, specialized algorithms were developed for the assignment problem. Of particular mention is the "Hungarian Method," a technique based upon a combinatorial theorem by the Hungarian mathematician Egerváry. Again, its importance as a solution method has diminished and we will not discuss it here, but relegate it to the website associated with this book.

Assignment problems have a number of applications, some not obviously related to assignments. The earliest story reported by G.B. Dantzig in his 1963 book on linear programming refers to it as the "marriage problem." The story is that a father has a number of daughters whom he wants to marry off. (It would work the

same way with sons, in case this is desired). There are a number of prospects for the matching, but each particular match requires a certain amount of dowry based on the (in-)compatibility of the couple. The thrifty father's overall objective to minimize the total amount of dowry he will have to pay.

The marriage story in its original form does not appear to be among the prime applications of assignment problems, though. Instead, decision makers may attempt to match items such as sports teams with the objective of maximizing the audience's appeal (and with it revenue, as appealing games—those with long-standing rivalries or those among teams with close standings—tend to attract larger crowds.)

There are some well-known and well-documented extensions to assignment problems, such as generalized assignment problems and quadratic assignment problems. Both types of extensions are not only very difficult from a computational point of view, but also beyond the scope of this volume.

Exercises

Problem 1 (production planning): Solve a variant of the standard production planning problem. Three products P_1, P_2, and P_3 are manufactured on two machines M_1 and M_2. Each of the products must be processed on both machines in arbitrary order. The unit profits of the products are \$18, \$12, and \$6, respectively, and the machine capacities are 24 and 16 hours per planning period. Table 2.2.16 indicates how many units of the products can be made each hour.

Table 2.2.16: Hourly production capabilities of the two machines

	P_1	P_2	P_3
M_1	3	5	10
M_2	6	4	12

In addition, it is required that at least ten units of the second product are made. Formulate a profit-maximizing linear programming problem.

Solution: Defining x_1, x_2, and x_3 as the quantities of the products to be made, the objective function below is formulated as usual. However, before we formulate the constraints, we have to adjust the units. The entries in Table 2.2.16 are expressed in terms of quantity units per hour. Multiplying them by the variables as usual would result in the meaningless units (quantity units)^2 per hour. Instead, we need to convert the entries in the table to hours per quantity units or, more conveniently, minutes per quantity unit. Multiplying by the variables (measured in quantity units), we obtain minutes used in the production, which can then be related to the capacities. The formulation can then be written as follows.

$$\begin{aligned} \text{Max } z = {} & 18x_1 + 12x_2 + 6x_3 \\ \text{s.t.} \quad & 20x_1 + 12x_2 + 6x_3 \leq 1{,}440 \\ & 10x_1 + 15x_2 + 5x_3 \leq 960 \\ & x_2 \geq 10 \\ & x_1, \quad x_2, \quad x_3 \geq 0. \end{aligned}$$

Incidentally, the optimal solution prescribes that 43.5, 10, and 75 units of the respective products are made for a total profit of \$1,353.

Problem 2 (allocation of time to courses): A student is planning the coming semester. In particular, he is attempting to allocate the weekly number of hours of study to the individual courses he is taking. Each hour of study will increase his mark by a certain quantity (starting at zero). Table 2.2.17 shows the marginal improvements of the marks given each hour of study (per week) as well as the marks required for passing the course.

Table 2.2.17: Input data for Problem 2

	Marketing	Organizational Behavior	Accounting	Operations Research	Finance
Marginal improvement of mark	5	4.5	5.5	3.5	5.5
Marks required for passing the course	50	55	60	50	50

For example, if our student were to allocate 15 hours (per week) to marketing, then his final mark is expected to be 15(5) = 75, which means passing the course.

The student's objective is to minimize the total number of hours studied. In addition, the following constraints have been identified:

- A passing grade should be achieved in *each course*.
- Obtain an average grade of at least 64.
- Suppose that the student has the option to flip hamburgers at McDonalds in his spare time. This job pays \$10 per hour. Assuming that the student has a total of 80 hours available for study and flipping, formulate that our student makes at least \$100 per week.
- The number of hours allocated to operations research should be at least 20 percent of the number of hours allocated to the other four subjects combined.

Solution: Given that time is the scarce resource, we define x_j as the number of hours allocated to studying the j-th subject, $j = 1, \ldots, 5$. The objective as well as the individual and overall passing requirements and the need to spend at least 20% of his studies on operations research are formulated in a straightforward fashion.

The need to make at least \$100 in the hamburger shop is formulated by first determining the hours used for hamburger flipping, which is the number of hours available overall (here 80) minus the hours used for studying (the sum of variables). These hours are then multiplied by the hourly wage of \$10, which is then the amount of money made. This amount is then required to be at least \$100. Another important—and frequently forgotten—part of this type of problem is the inclusion of constraints that limit the grades to 100. As a matter of fact, if these constraints were omitted, our student would aim for passing grades in most courses and allocate a large number of hours to one course, in which good marks are easy to obtain, so that he receives in excess of 100 marks. Clearly, this is not possible, making the additional limitations necessary.

$$\text{P: Min } z = x_1 + x_2 + x_3 + x_4 + x_5$$

$$\begin{aligned}
\text{s.t. } & 5x_1 && && && \geq 50 \\
& 4.5x_2 && && && \geq 55 \\
& && 5.5x_3 && && \geq 60 \\
& && && 3.5x_4 && \geq 50 \\
& && && && 5.5x_5 \geq 50
\end{aligned}$$

$$\begin{aligned}
& 5x_1 \leq 100 \\
& 4.5x_2 && && && \leq 100 \\
& && 5.5x_3 && && \leq 100 \\
& && && 3.5x_4 && \leq 100 \\
& && && && 5.5x_5 \leq 100
\end{aligned}$$

$$\begin{aligned}
& 5x_1 + 4.5x_2 + 5.5x_3 + 3.5x_4 + 5.5x_5 \geq 5(64) \\
& 10[80 - (x_1 + x_2 + x_3 + x_4 + x_5)] \geq 100 \\
& x_4 \geq .2(x_1 + x_2 + x_3 + x_5) \\
& x_1, x_2, x_3, x_4, x_5 \geq 0.
\end{aligned}$$

If the problem were solved, we find that the student studies a total of about 66½ hours and obtains minimum passing grades in Marketing, Organizational Behavior, and Operations Research, while he can expect 63.63 marks in Accounting (slightly better than the passing requirement of 60), and a 65 marks in Finance, significantly better than the required 50 marks.

Problem 3 (reformulation of the dynamic production-inventory problem): Formulate the problem in Section 2.2.5 with inventory variables that are defined at the end of the period.

Solution: Define now I_0, I_1, I_2, I_3, and I_4 as the inventory levels at the end of periods 0 (the beginning of the planning period), 1, 2, 3, and 4. We can then use the same formulation provided in Section 2.2.5, except that we need to replace I_t by I_{t-1} for $t = 1, 2, 3$, and 4. Doing so results in the formulation

P: Min $z = 1x_1 + 1.1x_2 + 1.2x_3 + 1.25x_4 + .05I_1 + .15I_2 + .15I_3$

s.t. $x_1 \leq 120$
$x_2 \leq 140$
$x_3 \leq 150$
$x_4 \leq 140$
$I_1 = I_0 + x_1 - 80$ (or, as $I_0 = 20$, $x_1 - I_1 = 60$)
$-I_2 + I_1 + x_2 = 70$
$-I_3 + I_2 + x_3 = 130$
$-I_4 + I_3 + x_4 = 150$ (or, as $I_4 = 0$, $x_4 + I_3 = 150$)
$x_1, x_2, x_3, x_4, I_1, I_2, I_3 \geq 0$.

Using an interpretation that reflects the inventory variables as defined here, the solution is again the same as before.

Problem 4 (a two-product production–inventory model): A firm manufactures two products. Their production capacities for the two products, unit production costs, and estimated demands are shown in Table 2.2.18.

Table 2.2.18: Parameters for Problem 4

	Month 1		Month 2		Month 3	
	Product *A*	Product *B*	Product *A*	Product *B*	Product *A*	Product *B*
Production capacity	70	40	80	30	80	10
Unit production cost	\$3.10	\$10.50	\$3.20	\$10.80	\$3.80	\$12.00
Estimated demand d_A,d_B	50	30	60	10	100	40

The opening inventories of the two products are 0 and 10 units, respectively. At the end of Month 3, we do not want any inventories left. The inventory carrying costs are 20¢ per unit of product *A* and 50¢ for each unit of product *B*. These costs are incurred whenever one unit of a product is carried over from one month to the next. The total inventory levels (for both products combined) from Month 1 to Month 2 should not exceed 40 units, while the total inventory level between Months 2 and 3 should not exceed 50 units. Find a cost-minimizing production plan.

Solution: The decision variables are x_{A1}, x_{A2}, and x_{A3} as the production quantities of product *A* in months 1, 2, and 3, and x_{B1}, x_{B2}, and x_{B3} as the production quantities of product *B* in the three months. In addition, the inventory levels at the beginning of periods 1, 2, 3, and 4 (where the inventory level at the beginning of

period 4 equals the inventory level at the end of period 3) for the two products are defined as I_{A1}, I_{A2}, I_{A3}, and I_{A4}, and I_{B1}, I_{B2}, I_{B3}, and I_{B4}, respectively.

The formulation of the problem is then

$$
\begin{aligned}
&\text{P: Min } z = 3.1x_{A1} + 3.2x_{A2} + 3.8x_{A3} + 10.5x_{B1} + 10.8x_{B2} + 12x_{B3} \\
&\qquad + .2I_{A2} + .2I_{A3} + .5I_{B2} + .5I_{B3} \\
&\text{s.t. } x_{A1} \le 70 \\
&x_{A2} \le 80 \\
&x_{A3} \le 80 \\
&x_{B1} \le 40 \\
&x_{B2} \le 30 \\
&x_{B3} \le 10 \\
&I_{A1} = 0 \\
&I_{B1} = 10 \\
&I_{A2} = I_{A1} + x_{A1} - 50 \\
&I_{A3} = I_{A2} + x_{A2} - 60 \\
&I_{A4} = I_{A3} + x_{A3} - 100 \\
&I_{B2} = I_{B1} + x_{B1} - 30 \\
&I_{B3} = I_{B2} + x_{B2} - 10 \\
&I_{B4} = I_{B3} + x_{B3} - 40 \\
&I_{A4} = 0 \\
&I_{B4} = 0 \\
&I_{A2} + I_{B2} \le 40 \\
&I_{A3} + I_{B3} \le 50 \\
&x_{A1}, x_{A2}, x_{A3}, x_{B1}, x_{B2}, x_{B3}, I_{A1}, I_{A2}, I_{A3}, I_{A4}, I_{B1}, I_{B2}, I_{B3}, I_{B4} \ge 0.
\end{aligned}
$$

The optimal solution is shown in Table 2.2.19. The associated costs are \$1,498.

Table 2.2.19: Optimal solution of Problem 4

	Month 1			Month 2			Month 3			
P_1	I_{A1}	x_{A1}	d_{A1}	I_{A2}	x_{A2}	d_{A2}	I_{A3}	x_{A3}	d_{A3}	I_{A4}
	0	+50	−50	=0			20 + 80		−100	=0
				0	+80	−60	=20			
P_2	I_{B1}	x_{B1}	d_{B1}	I_{B2}	x_{B2}	d_{B2}	I_{B3}	x_{B3}	d_{B3}	I_{B4}
	10	+ 30	− 30	= 10			30 + 10		− 40	= 0
				10 + 30 − 10			=30			

Problem 5 (blending of tobaccos): The buyer of a large tobacco manufacturer has the choice of four tobacco types *Virginia*, *Burley*, *Latakia*, and *Kentucky*. Once sufficient quantities have been purchased, they will make three blends of pipe tobacco, *viz*., *Sweet Smell*, *Brown Lung*, and *Black Death*. The following information is available.

- The four types of tobacco cost \$3, \$6, \$5, and \$2 per pound (in the order they were mentioned).
- The final blends are sold by the 4 oz pouch, i.e. there are four pouches per pound.
- The blends sell for \$7, \$9, and \$12 per pouch (in the above order).
- *Sweet Smell* consists of 20% *Virginia*, 50% *Burley*, and 30% *Latakia*, *Brown Lung* is blended from 40% *Latakia* and equal proportions of the remaining tobaccos, and *Black Death* is 80% *Kentucky* and 20% *Latakia*.
- The four tobaccos are available in limited quantities. We may purchase up to 300 lbs of *Virginia*, 500 lbs of *Burley*, 100 lbs of *Latakia*, and 50 lbs of *Kentucky*.
- Our customers have placed orders for exactly 500 pouches of *Sweet Smell* and 400 pouches of *Brown Lung*. There are no firm order for the expensive *Black Death*, but we expect to be able to sell between 80 and 120 pouches.

(a) Formulate a linear programming problem for the above situation. Define the variables clearly.
(b) Assume that there is a 5% loss in the blending process. Explain the changes in the formulation.

Solution: (a) As usual, the variables are denoted by x_{ij} and defined as the quantity of i-th raw tobacco in j-th blend. The problem is very similar to that in Section 2.2.6. The only major difference is that the raw materials are measured in pounds, while the products are sold by the pouch. As four pouches make a pound, we need to convert pouches to pounds by multiplying the quantities of the products by 4. The problem can then be formulated as:

$$\begin{aligned}\text{P: Max } z = {} & 7(x_{11} + x_{21} + x_{31} + x_{41})4 + 9(x_{12} + x_{22} + x_{32} + x_{42})4 \\ & + 12(x_{13} + x_{23} + x_{33} + x_{43})4 - 3(x_{11} + x_{12} + x_{13}) - 6(x_{21} + x_{22} + x_{23}) \\ & - 5(x_{31} + x_{32} + x_{33}) - 2(x_{41} + x_{42} + x_{43})\end{aligned}$$

$$\begin{aligned}\text{s.t. } & x_{11} + x_{12} + x_{13} \le 300 \\ & x_{21} + x_{22} + x_{23} \le 500 \\ & x_{31} + x_{32} + x_{33} \le 100 \\ & x_{41} + x_{42} + x_{43} \le 50\end{aligned}$$

$$\begin{aligned}& 4(x_{11} + x_{21} + x_{31} + x_{41}) = 500 \\ & 4(x_{12} + x_{22} + x_{32} + x_{42}) = 400 \\ & 4(x_{13} + x_{23} + x_{33} + x_{43}) \le 120 \\ & 4(x_{13} + x_{23} + x_{33} + x_{43}) \ge 80\end{aligned}$$

$$\begin{aligned}& x_{11} = 0.2(x_{11} + x_{21} + x_{31} + x_{41}) \\ & x_{21} = 0.5(x_{11} + x_{21} + x_{31} + x_{41}) \\ & x_{31} = 0.3(x_{11} + x_{21} + x_{31} + x_{41}) \\ & x_{12} = 0.2(x_{12} + x_{22} + x_{32} + x_{42}) \\ & x_{22} = 0.2(x_{12} + x_{22} + x_{32} + x_{42})\end{aligned}$$

$$x_{32} = 0.4(x_{12} + x_{22} + x_{32} + x_{42})$$
$$x_{42} = 0.2(x_{12} + x_{22} + x_{32} + x_{42})$$
$$x_{33} = 0.2(x_{13} + x_{23} + x_{33} + x_{43})$$
$$x_{43} = 0.8(x_{13} + x_{23} + x_{33} + x_{43})$$

$$x_{11}, x_{12}, x_{13}, x_{21}, x_{22}, x_{23}, x_{31}, x_{32}, x_{33}, x_{41}, x_{42}, x_{43} \geq 0.$$

(b) Suppose that there is a 5% loss in the blending process. This can easily be accounted for as follows. Everywhere the quantity of a product is referred to, it is replaced by the quantity multiplied by 1 – 5% = 0,95. In this formulation, we replace $(x_{11} + x_{21} + x_{31} + x_{41})$ by $(x_{11} + x_{21} + x_{31} + x_{41})(.95)$ and similar for $(x_{12} + x_{22} + x_{32} + x_{42})$ and $(x_{13} + x_{23} + x_{33} + x_{43})$ in the objective function, the second set of constraints, and the right-hand sides of the last set of constraints. It would also be easy to include different losses for different products.

Problem 6 (blending of gasolines): Table 2.2.20 describes components in a petroleum refinery that can be used to blend gasoline.

Table 2.2.20: Input data for Problem 6

Component	Availability (in barrels)	Octane number	Vapor pressure	Cost per barrel
Naphta	30,000	85	10	\$53
Hydrocrackate	45,000	79	4	\$48
Reformate	20,000	101	9	\$62
Alkylate	15,000	108	5	\$69

The purpose is to blend two types of gasoline, *Regular* and *Premium*, so as to minimize the overall costs required to satisfy the demand. The *Regular* brand consists of *Naphta*, *Hydrocrackate*, and *Reformate*, while *Premium* consists of *Naphta*, *Hydrocrackate*, and *Alkylate*. The total contracted demand for gasoline is 80,000 barrels.

- *Regular*: The octane number must be at least 87 and the vapor pressure cannot exceed 7.2.
- *Premium*: The octane number must be at least 91 and the vapor pressure cannot exceed 6.8.

Assume that there are no losses in the blending process and that all quantities blend linearly by quantity.

Solution: Given the four "raw materials" *Naphta*, *Hydrocrackate*, *Reformate*, and *Alkylate* along with the two products *Regular* and *Premium*, we can define the variables x_{ij} as the quantity of raw material i in product j.

$$\text{Min } z = 53(x_{11} + x_{12}) + 48(x_{21} + x_{22}) + 62x_{31} + 69x_{42}$$

$$\begin{aligned}
&\text{s.t. } x_{11} + x_{12} \le 30{,}000 && \text{(availability of } \textit{Naphta}\text{)}\\
&x_{21} + x_{22} \le 45{,}000 && \text{(availability } \textit{Hydrocrackate}\text{)}\\
&x_{31} \le 20{,}000 && \text{(availability of } \textit{Reformate}\text{)}\\
&x_{42} \le 15{,}000 && \text{(availability of } \textit{Alkylate}\text{)}
\end{aligned}$$

$$x_{11} + x_{12} + x_{21} + x_{22} + x_{31} + x_{42} = 80{,}000 \qquad \text{(demand)}$$

$$\begin{aligned}
&85x_{11} + 79x_{21} + 101x_{31} \ge 87(x_{11} + x_{21} + x_{31}) && \text{(octane } \textit{Regular}\text{)}\\
&10x_{11} + 4x_{21} + 9x_{31} \le 7.2(x_{11} + x_{21} + x_{31}) && \text{(vapor pressure } \textit{Regular}\text{)}
\end{aligned}$$

$$\begin{aligned}
&85x_{12} + 79x_{22} + 108x_{42} \ge 91(x_{12} + x_{22} + x_{42}) && \text{(octane } \textit{Premium}\text{)}\\
&10x_{12} + 4x_{22} + 5x_{42} \le 6.8\,(x_{12} + x_{22} + x_{42}) && \text{(vapor pressure } \textit{Premium}\text{)}
\end{aligned}$$

$$x_{11}, x_{21}, x_{31}, x_{12}, x_{22}, x_{42} \ge 0.$$

Problem 7 (blending with exact requirements): A fish processing plant makes two types of fish sticks, the *Scrumptious Skipper* and the *Delicious Sailor*. The *Skipper* consists of exactly 30% pollock, 40% haddock, and 30% sole, while the *Sailor* contains 30% pollock, 20% haddock, and 50% sole. A one-pound package of the *Skipper* sells for \$2.50, while a one-pound pack of the *Sailor* retails for \$3.50. There are 4,000 lbs of pollock, 3,000 lbs of haddock, and 3,000 lbs of sole available in the plant. Formulate a profit-maximizing linear program.

Solution: The usual thought would be to define the variables x_{ij} as the quantity of the i-th type of fish in the j-th type of fish sticks. The formulation is then

$$\begin{aligned}
&\text{P: Max } z = 2.5(x_{11} + x_{21} + x_{31}) + 3.5(x_{12} + x_{22} + x_{32})\\
&\text{s.t. } x_{11} + x_{12} \le 4{,}000\\
&x_{21} + x_{22} \le 3{,}000\\
&x_{31} + x_{32} \le 3{,}000\\
&x_{11} = .3(x_{11} + x_{21} + x_{31})\\
&x_{21} = .4(x_{11} + x_{21} + x_{31})\\
&x_{31} = .3(x_{11} + x_{21} + x_{31})\\
&x_{12} = .3(x_{12} + x_{22} + x_{32})\\
&x_{22} = .2(x_{12} + x_{22} + x_{32})\\
&x_{32} = .5(x_{12} + x_{22} + x_{32})\\
&x_{11}, x_{12}, x_{21}, x_{22}, x_{31}, x_{32} \ge 0.
\end{aligned}$$

While this is a correct formulation, it is too large for what it does. We could simply formulate variables x_1 and x_2 as the quantities of the two fish stick packages that we make, and formulate

$$
\begin{aligned}
&\text{Max } z = 2.5x_1 + 3.5x_2 \\
&\text{s.t.} \quad .3x_1 + .3x_2 \le 4{,}000 \\
&\qquad .4x_1 + .2x_2 \le 3{,}000 \\
&\qquad .3x_1 + .5x_2 \le 3{,}000 \\
&\qquad x_1, \quad x_2 \ge 0.
\end{aligned}
$$

In both cases, we make 6,428.57 packages of the *Skipper* and 2,142.86 packs of the *Sailor* for a profit of $23,571.43. This shorter formulation is possible as there is a fixed relation between the number of packages of the two products and the quantity of the fish input (e.g., the quantity of pollock in the packages of *Skipper* is exactly 0.3 times the quantity of *Skipper* packages). This was not the case in Problems 5 and 6.

Problem 8 (a transportation problem): Consider a school district's problem to assign student from different villages to central schools. Typically, with the closing of small neighborhood schools of the "little red schoolhouse" type and the establishment of larger centralized schools, it has become necessary to bus the students to the schools, (as these distances would even make Abe Lincoln take the bus). The objective is to ensure that all students must be able to take a bus, and school capacities cannot be violated.

Suppose there are three villages with 30, 50, and 20 students each. The two centralized schools have capacities of 70 and 60 students, respectively. The distances between the villages and the schools are shown in the matrix **C** below.

$$
\mathbf{C} = \begin{bmatrix} 20 & 15 \\ 40 & 30 \\ 60 & 20 \end{bmatrix}.
$$

(a) Formulate the problem.
(b) Suppose that the buses available to the district each have a capacity of 35 students. Formulate constraints to ensure that there are no overfilled buses.
(c) In addition to the constraints under (b), it is now required that each school is filled to at least 75% capacity.

Solution: (a) We first define variables, so that x_{ij} denotes the number of students bused from village i to school j. The model can then be formulated as follows.

$$
\text{P: Min } z = 20x_{11} + 15x_{12} + 40x_{21} + 30x_{22} + 60x_{31} + 20x_{32}
$$

$$
\begin{aligned}
\text{s.t.} \quad & x_{11} + x_{12} = 30 \\
& x_{21} + x_{22} = 50 \\
& x_{31} + x_{32} = 20 \\
& x_{11} + x_{21} + x_{31} \le 70 \\
& x_{12} + x_{22} + x_{32} \le 60 \\
& x_{11}, \; x_{12}, \; x_{21}, \; x_{22}, \; x_{31}, \; x_{32} \ge 0.
\end{aligned}
$$

Incidentally, the solution is summarized in the optimal transportation plan

$$\overline{\mathbf{T}} = \begin{bmatrix} 30 & 0 \\ 10 & 40 \\ 0 & 20 \end{bmatrix}.$$

The total mileage to bus all students to schools is 2,600, and while the second school is filled to capacity, the first students houses only 40 students, well shy of its capacity of 70.

(b) The most obvious way to formulate this constraint is to impose capacities on all routes, i.e., write six additional constraints $x_{11} \leq 35$, $x_{12} \leq 35$, …, $x_{32} \leq 35$. This is, however, unnecessary, as only buses leading out of the second village could possibly have more students than the bus capacity allows. Hence it is sufficient to add the two constraints $x_{21} \leq 35$ and $x_2 \leq 35$. The new solution has the transportation plan

$$\overline{\mathbf{T}} = \begin{bmatrix} 25 & 5 \\ 15 & 35 \\ 0 & 20 \end{bmatrix}$$

requiring a total mileage of 2,625, a very minor increase from the original 2,600.

(c) Filling the schools to at least 75% of capacity requires the two schools to house at least 70(.75) = 52.5 and 60(.75) = 45 students. Since integrality is required and these are lower bounds on the number of students, we have to round up the first number to 53. We then add the constraints

$$x_{11} + x_{21} + x_{31} \geq 53$$
$$x_{12} + x_{22} + x_{32} \geq 45$$

to the problem. The optimal solution is then shown in the transportation plan

$$\overline{\mathbf{T}} = \begin{bmatrix} 30 & 0 \\ 23 & 27 \\ 0 & 20 \end{bmatrix}$$

with an associated total mileage of 2,730.

2.3 Graphical Representation and Solution

Having discussed a variety of different applications of linear programming problems, this section will first demonstrate how linear programming problems can be represented graphically. We then discuss in some detail a graphical solution method. This is followed by a discussion of a number of special cases that may occur in the modeling and solution process, and how to react to them as an analyst.

Throughout this chapter, we will restrict ourselves to the case of two variables in order to accommodate easy graphing. This does, of course, mean that the technique we describe in this section is not made for the solution of realistic problems that typically have tens of thousands of variables. Much rather, the purpose of this discussion is to create an understanding of what happens in the solution of linear programming problems and what the difficulties are, regardless of the size of the problem.

The first subsection will demonstrate how constraints and objective functions can be graphed and how the problem can be solved graphically. Based on the understanding of this material, Section 2.3.2 will then discuss a number of special situations that may occur in the solution process and how to deal with the resulting messages from the solver.

2.3.1 The Graphical Solution Method

As discussed in the introduction to linear programming, each model consists of an objective function and a number of constraints. This subsection will first demonstrate how to plot constraints, then show how to deal with objective functions, and then put it all together in the graphical solution method.

Assuming that we have two variables x_1 and x_2, a constraint could be a linear function such as $3x_1 + 2x_2 \leq 6$. In order to plot this constraint, it is easiest to first consider the associated equation $3x_1 + 2x_2 = 6$. It is known that the line in two dimensions is uniquely determined by two points. In order to do so, we can simply set either of the variables to any value we like and solve for the other variable, resulting in one of the required points. Repeating this step with a different value will result in a second point. The straight line that leads through both of these points is then the set of all points that satisfy the equation.

In our example, setting $x_1 = 0$ leads to $2x_2 = 6$ or $x_2 = 3$, so that the first point is $(x_1, x_2) = (0, 3)$. The second point can be obtained by setting $x_2 = 0$, which leads directly to $3x_1 = 6$, or, equivalently, $x_1 = 2$. As a result, our second point is $(x_1, x_2) = (2, 0)$. The straight line in Figure 2.3.1 is the set of points that satisfy $3x_1 + 2x_2 = 6$.

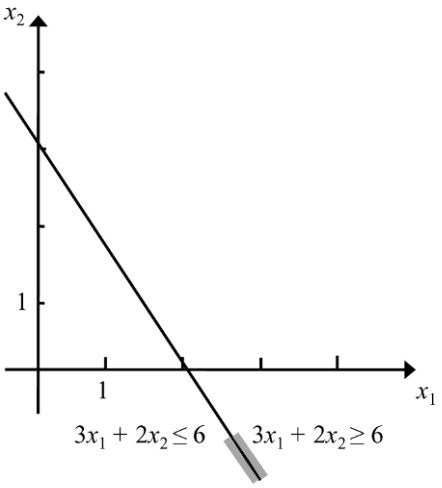


Figure 2.3.1

So far, we have determined that an equation is represented in two dimensions as a straight line. (Note that in a single dimension an equation is just a point). In three dimensions, an equation would be represented by a plane, so that in general, we speak about an equation in any number of dimensions being represented by a *hyperplane*.

Back in our two-dimensional example, recall that the constraint in question is the inequality $3x_1 + 2x_2 \leq 6$, so that not only the set of points on the line are addresses. As a matter of fact, a $\leq$ or $\geq$ inequality will refer to the set of points on the line and all points in one of the two *halfplanes* generated by the line. The obvious question is then which of the two halfplanes is addressed by the constraint in question. Some people might believe that the halfplanes that belong to $\leq$ inequalities are below the line, while those of $\geq$ are above the line. This is not true, as each $\leq$ inequality can be rewritten as an equivalent $\geq$ inequality. In our example, the constraint $3x_1 + 2x_2 \leq 6$ is equivalent to its counterpart $-3x_1 - 2x_2 \geq -6$. Both constraints define exactly the same set of points.

A simple way to determine the proper halfplane is to choose any point that is not located on the line we have plotted and determine whether or not it satisfies the constraint in question. If so, then the point is located on the proper side of the line, otherwise the halfplane is on the other side. In our example, consider, for instance, the origin as a point. Its coordinates are (0, 0), so that the constraint $3x_1 + 2x_2 \leq 6$ reduces to $0 \leq 6$, which is correct. This means that the origin is on the "correct" side of the line, which allows us to determine the halfplane as being on the lower left side of the line. Had we chosen the point, say, (4, 2) instead, our constraint would have been $3(4) + 2(2) \leq 6$ or $16 \leq 6$, which is wrong, meaning that the point

(4, 2) is located on the "wrong" side of the line. As a matter of fact, the set of points on the line and in the halfplane to the upper right of the line is determined by the constraint $3x_1 + 2x_2 \geq 6$. Figure 2.3.1 shows both the hyperplane and both halfplanes for all three types of constraints allowed in linear programming: $=$, $\leq$, and $\geq$.

Given that an equation is represented by a straight line in two and a plane in three dimensions, an inequality is represented by a halfplane in two dimensions and half the space in three dimensions (the separating line is given by the associated equation and the other half of the space is defined by the same inequality but with inverted inequality sign). This has led to the term *halfspace* that, in contrast to halfplane, which applies only to two dimensions, applies to the representation of an inequality of the $\leq$ or $\geq$ type in any number of dimensions.

At this point, we are able to plot the hyperplane or halfspace for each constraint in a given problem. In order to determine the *feasible set* (also called *feasible region* or *set of feasible solutions*), we first need to define a solution as feasible, if it satisfies *all* of the given constraints. This is not really a restriction, as if we do not want a constraint to be satisfied, why include it in the problem in the first place? Given that all constraints must be satisfied, the feasible set is then the intersection of the halfspaces and/or hyperplanes that correspond to all constraints of the problem.

As a numerical illustration, consider the following numerical example.

$$
\begin{array}{lrcll}
\text{P: Max } z = & 2x_1 + & 3x_2 & & \\
\text{s.t.} & x_1 + & 4x_2 \leq & 12 & (I) \\
& 2x_1 - & x_2 \geq & -2 & (II) \\
& 5x_1 + & 3x_2 \leq & 15 & (III) \\
& 4x_1 + & 6x_2 \geq & 6 & (IV) \\
& x_1 & \geq & 0 & (V) \\
& & x_2 \geq & 0. & (VI)
\end{array}
$$

The feasible set determined by the constraints of problem P, is shown as the shaded area in Figure 2.3.2.

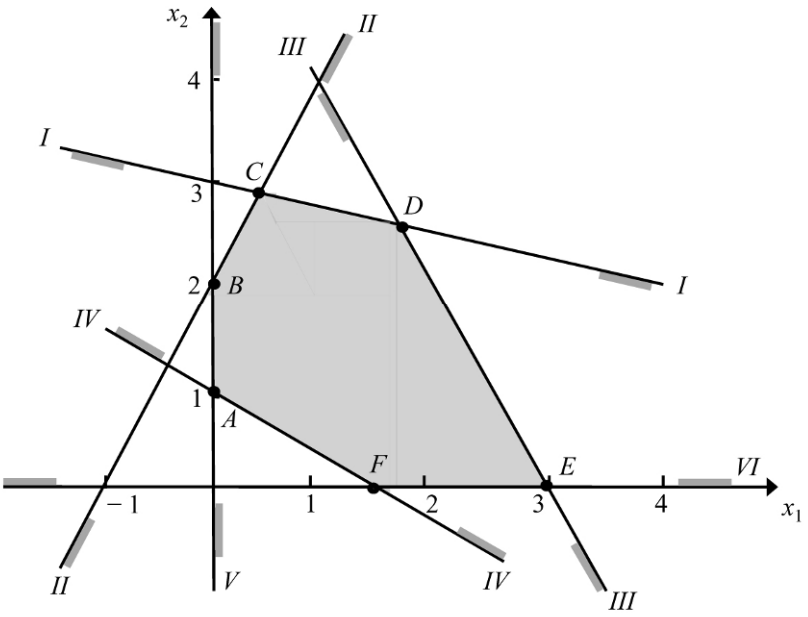


Figure 2.3.2

In our two-dimensional space, the feasible set is a linearly bounded polygon (in general, referred to as a *polytope*). It consists of the boundary with its linear segments and *corner points* (frequently referred to as *extreme points*) A, B, …, F, as well as the interior. At each of the extreme points, at least two constraints are satisfied as equations. These constraints are usually referred to as *binding* (or *tight*) *at this point*. In our example, at point A, the constraints *IV* and *V* are satisfied as equations, at point B, constraints *II* and *V* are satisfied as equations, at point C, constraints *I* and *II* are satisfied as equations, and so forth.

We can then determine the exact coordinates of all extreme points by solving a system of simultaneous linear equations. For instance, for point A, the system of simultaneous linear equations includes all constraints satisfied as equations at this point. Here, these are the equations based on constraints *IV* and *V*, so that the system is

$$\begin{aligned} 4x_1 + 6x_2 &= 6 \text{ and} \\ x_1 \quad\quad &= 0. \end{aligned}$$

Replacing $x_1 = 0$ in the first equation and solving for x_2, we obtain $x_2 = 1$, so that the coordinates are $(x_1, x_2) = (0, 1)$.

Similarly, consider point *C*. At his point, the constraints *I* and *II* are binding, so that we have the set of simultaneous linear equations

$$x_1 + 4x_2 = 12$$
$$2x_1 - x_2 = -2.$$

A system like this can be solved by any of the pertinent methods, see Appendix C of this volume. One (albeit somewhat awkward) possibility is to use the *substitution technique*. Here, we solve the first equation for x_1, resulting in $x_1 = 12 - 4x_2$. We then replace x_1 by this expression in the second equation, so that we obtain $2(12 - 4x_2) - x_2 = -2$. Solving this equation for x_2 results in $x_2 = \frac{26}{9} \approx 2.8889$. Replacing x_2 by this value in $x_1 = 12 - 4x_2$ and solving for x_1 results in $x_1 = \frac{4}{9} \approx 0.4444$. Table 2.3.1 shows the points, the constraints that are satisfied as equations at that point, and their exact coordinates.

Table 2.3.1: Coordinates of extreme points of sample problem

Point	Constraints binding at point	Coordinates (x_1, x_2)	z-value
A	*IV, V*	(0, 1)	3
B	*II, V*	(0, 2)	6
C	*I, II*	$(\frac{4}{9}, 2\frac{8}{9}) \approx (.4444, 2.8889)$	9.5556
D	*I, III*	$(1\frac{7}{17}, 2\frac{11}{17}) \approx (1.4118, 2.6471)$	10.7649
E	*III, VI*	(3, 0)	6
F	*IV, VI*	(1½, 0)	3

In n dimensions, each extreme point is determined by the intersection of n hyperplanes, so that we have to solve a system of simultaneous linear equations in n variables with n equations to determine the coordinates for each of these extreme points.

Consider now the objective function. To simplify matters, we will at first ignore the constraints and deal exclusively with the objective function and its representation, before we combine objective function and constraints in the graphical solution method.

For now, consider the objective function Max $z = 2x_1 + 5x_2$. Ignoring the maximization for a moment, we have $2x_1 + 5x_2 = z$, which is nothing but a regular constraint with an unknown right-hand side value z. As discussed above, for any value of z, we have an equation that can be represented by a hyperplane in the space of variables, here the (x_1, x_2) space. Figure 2.3.3 shows these lines for values of $z = 5$, 10, 15, and 20.

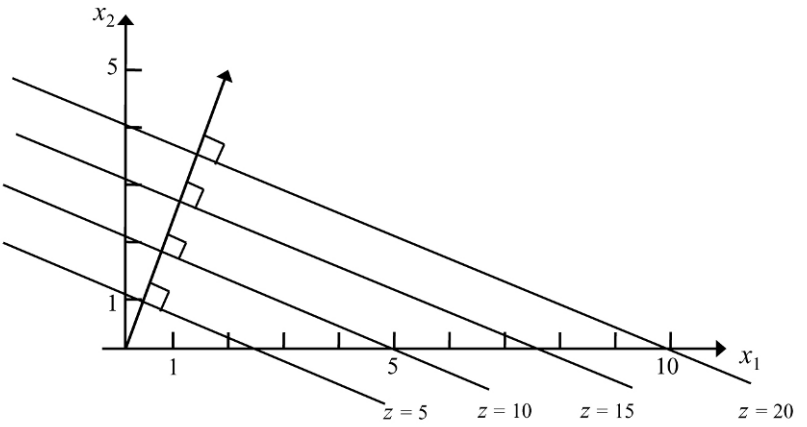


Figure 2.3.3

Depending on the type of objective function under consideration, these lines are usually referred to as *iso-profit lines*, *iso-cost lines*, or simply *contour lines*. Their name derives from the fact that all points on any one of these lines have the same value of the objective function. In other words, given the objective function under consideration, all points on the line labeled $z = 5$ are considered equally good by the decision maker. Similar, a decision maker will consider all points on the line $z = 10$ as equally good—but better than those on the $z = 5$ line. The value of the objective function gets better in the northeasterly direction.

It is then possible to construct a vector that points into the direction, in which the objective function improves. This is the so-called *gradient of the objective function*. Formally, a gradient is the vector of partial derivatives, but here it is sufficient to think of it as the direction in which the solutions get better. The gradient is constructed as follows, where we use gain our numerical example with the objective Max $z = 2x_1 + 5x_2$. Each term of the objective function can be thought of as the product of the step direction and the step length. Here, x_1 means move to the right, and the coefficient 2 tells us to move two steps into that direction. The next term indicates that we should move 5 steps into the x_2 direction. Starting at an arbitrary point, we first move 2 steps into the x_1 direction, followed by 5 steps into the x_2 direction. The starting point is then connected to the end point, resulting in the gradient. Usually, we start these moves at the origin, but this is not necessary.

Observe that the gradient of the objective function is perpendicular to the iso-profit lines. Once we have the gradient, it is not necessary to explicitly plot any of the iso-profit lines. (In more than two dimensions, the gradient is a ray that is orthogonal—the generalization of perpendicular to n dimensions—to the iso-profit hyperplanes). From a practical point of view, we can plot the gradient of the objective function and then push the perpendicular iso-profit lines as much into its direction as possible—the farther we push, the higher the profit.

Before putting it all together and describing the graphical solution technique, some properties of the objective function are worth mentioning. Suppose that in the above objective function each of the terms is measured in dollars. Assume now that we have decided to measure the profit in Indian rupees instead. Suppose that the exchange rate is 50 rupees per dollar, so that the objective function is now Max $z' = 100x_1 + 250x_2$. Plotting this objective, we find that while the gradient is much longer, the direction of the objective function is exactly the same. As we will see later, such a change of currency will result in exactly the same solution as the original objective function, only the objective value changes: z' will be 50 times the value z.

Another point of interest concerns minimization functions. What if the objective function minimizes some costs, e.g., Min $z = 3x_1 + 7x_2$? No special procedure is needed, as we can simply transform the minimization objective into an equivalent maximization objective by multiplying it by a negative number, e.g., (−1). This will result in the equivalent objective Max $-z = -3x_1 - 7x_2$. As far as the gradient of this function is concerned, it leads from the origin −3 steps into the x_1 direction (i.e., three steps to the left), followed by −7 steps into the x_2 direction (i.e., 7 steps down). Everything else remains exactly the same, the value of the objective function improves (i.e., gets smaller in case of a minimization function) as we shift the iso-cost lines more and more into the direction of the gradient. Figure 2.3.4 shows the gradients for the following objective functions:

(a) Max $z_1 = 4x_1 - 3x_2$
(b) Max $z_2 = -x_1 + 3x_2$
(c) Min $z_3 = -2x_1 - x_2$
(d) Min $z_4 = 2x_1 + 3x_2$

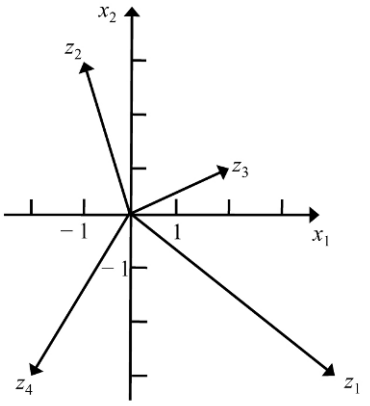


Figure 2.3.4

It is worthwhile to notice that if we have one function such as $z_1 = 4x_1 - 3x_2$ in the above example, maximizing the function leads to a gradient that points into a southeasterly direction. Minimizing the same function leads into the northwest, diametrically opposed to the maximization of the same function.

We are now able to describe the complete graphical solution technique. After determining the feasible set, we plot the gradient of the objective function and move its iso-profit lines into the direction of the gradient, until we hit the last feasible point. While there are solutions with better objective function values beyond this point, none of them is feasible. Thus the last feasible point into the direction of the gradient is the optimal point.

It is apparent that in this procedure, points in the interior of the feasible set cannot be optimal; any optimal solution will be on the boundary of the feasible set. In fact, Dantzig has proved his famous corner point theorem, which we will state here formally.

Theorem (Corner point theorem, Dantzig): At least one optimal solution is located at an extreme point of the feasible set.

The graphical solution method will identify such a corner point, whose exact coordinates we will have to determine next. As demonstrated earlier in this section, this is done by way of solving a system of simultaneous linear equations. Once the exact coordinates of the optimal solution point have been determined, all that is left to do is to determine the quality of the solution, as measured by the objective function. This is accomplished by replacing the variables in the objective function by their optimal values and thus computing the z-value.

We can summarize the procedure in the following steps:

(1) Graph the constraints and determine the set of feasible solutions.
(2) Plot the gradient of the objective function.
(3) Apply the graphical solution technique that pushes iso-profit lines into the direction of the gradient until the last feasible point is reached. This is the optimal solution $\bar{\mathbf{x}}$.
(4) Determine which constraints are satisfied as equations at $\bar{\mathbf{x}} = (\bar{x}_1, \bar{x}_2)$. Write them as equations and solve the resulting system of simultaneous linear equations for the exact coordinates of the optimal solution.
(5) Use the coordinates of the optimal point in the objective functions and compute the value of the objective function.

Applying the first two steps of this procedure to the problem stated in the beginning of this subsection, we obtain the graph in Figure 2.3.5. Pushing now the iso-profit lines (some of which are shown) into the direction of the gradient, we find that the last feasible point on our way into a Northeasterly direction id the extreme point *D*. This is the optimal point. The constraints *I* and *III* are binding at

this point, so that we solve the system of simultaneous linear equations that consists of relations *I* and *III* written as equations, i.e.,

$$x_1 + 4x_2 = 12$$
$$5x_1 + 3x_2 = 15.$$

The optimal solution is $(\bar{x}_1, \bar{x}_2) = (1\frac{7}{17}, 2\frac{11}{17}) \approx (1.4118, 2.6471)$, and the associated value of the objective function is $\bar{z} = 10\frac{13}{17} \approx 10.7647$. It can be shown that this solution is not only optimal, but is the *unique optimal solution* to the problem. Sometimes, an optimal point is found, such that at least one of its neighboring extreme points has the same value of the objective function, and as such is also optimal. This happens in our problem, if the same objective function were minimized rather than maximized. In this case, we would find the points *A* and *F* both as optimal solution points with $\bar{z} = 3$. More about this issue can be found in the next subsection on special cases.

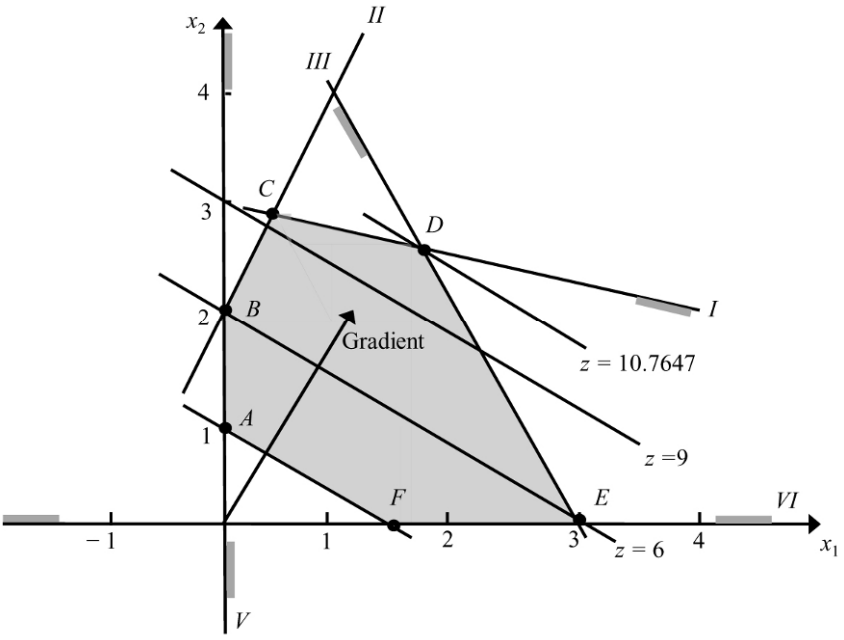


Figure 2.3.5

While the graphical method as demonstrated above is an exact method (no approximations were made in the process), its use is to explain the main concepts and difficulties involved in solving linear programming problems. The reason is that practical problems have not two or three, but tens of thousands of variables, making graphing impossible. Since the graphical solution technique uses the exact

pictorial knowledge of the feasible set, it will be necessary to find an algebraic technique that is independent of the graphical image. Dantzig's *simplex method* is such a tool. Rather than moving through the feasible space directly to the optimal solution, the simplex method is an *incremental technique* that starts with a feasible solution (which can be determined by some technique), improves it, tests whether or not an optimal solution has been found, and if not, increases the solution further. It does so by moving on the boundary of the feasible set from one extreme point to an adjacent extreme point. The method also belongs to the class of *feasible* (and improving) *direction methods*. This means that a step from a (feasible) extreme point to an adjacent extreme point is only made, if the solution remains feasible, and the value of the objective function improves in the process. A feature of the feasible set, called *convexity*, guarantees that if a point is found none of whose neighbors has a better z-value than the one we are presently at, this is an overall (i.e., global) optimal solution.

To demonstrate a simplex path, i.e., the sequence of extreme points generated and examined by the simplex method, consider again the example of Figure 2.3.5 and assume that we have "somehow" determined point A as a starting point. Point A has two neighboring extreme points F and B. Both are feasible, so that moves are possible. However, while the move from A to B improves the value of the objective function as B is on a higher iso-profit line, the move from A to F will leave the value of the objective function unchanged. (This is one of the special cases discussed in the next subsection). Since we are looking for improvements, the simplex method will move to point B. At that point, we have again two neighboring extreme points, *viz*., A and C. While a move from B to A retains feasibility of the solution, the z-value would decrease, disallowing such move. On the other hand, moving from B to C maintains feasibility and improves the value of the objective function, so that the simplex method makes this move. At point C, we have again two neighbors, which are B and D. Moving to B is not allowed, as this would decrease the z-value. On the other hand, a move to D not only maintains feasibility, but also increases the value of the objective function. The neighboring extreme points at point D are C and E. Moving either way will keep the solution feasible, but in both cases the value of the objective function will decrease. At this point, the method terminates with the message that point D is an optimal solution.

While examples have been constructed in which the simplex algorithm performs very poorly, the average performance of the algorithm has been excellent. Given a problem with m constraints, there is consensus that on average, the simplex algorithm needs to examine only $1½m$ extreme points. In each step, we need to examine an extreme point, which means we must solve a system of simultaneous linear equations. Traditionally, computational details of this method, which is considered to be one of the ten top algorithms of the 20th century, have been included in texts such as this. Given the abundance of software (some of it even free) and the fact that users do not need to know details about how the method functions, we will not discuss the method in this book. Instead, for details

interested readers are referred to Eiselt and Sandblom (2007) and the website that accompanies this book.

Finally, we would like to address the question why, given the tremendous computing power of today's equipment, we do not simply enumerate all extreme points, determine their exact coordinates and their objective values, and then choose the one with the best objective value (meaning the highest value for maximization and lowest value for minimization problems), which then will be the optimal solution. Given Dantzig's corner point theorem, the procedure is certainly valid in that it will find an optimal solution. However, as the example below will clearly demonstrate, it is of no practical value.

As an example consider a problem whose constraints are $0 \leq x_1 \leq 1$, $0 \leq x_2 \leq 1$, and so forth for all n variables. With two variables, the feasible set is a square with the four corner points $(x_1, x_2) = (0, 0)$, $(1, 0)$, $(1, 1)$, and $(0, 1)$. With three variables, the set is a unit cube with eight corner points $(x_1, x_2, x_3) = (0, 0, 0)$, $(0, 0, 1)$, $(0, 1, 0)$, $(1, 0, 0)$, $(0, 1, 1)$, $(1, 0, 1)$, $(1, 1, 0)$, and $(1, 1, 1)$. Continuing in this fashion, we will have a feasible set in the shape of a hypercube with 2^n extreme points. Given a very small practical problem with only $n = 100$ variables, there are $2^{100} \approx 10^{30}$ corner points. Given the fastest machines today that can deal with 10^{15} floating point operations per second (called 1 petaflop), and assuming that one such operation can determine one extreme point (it cannot: we would need to solve a system of simultaneous linear equation with 100 variables and 100 equations), we would still need close to 670,000 years to solve the problem. This shows the uselessness of enumeration techniques for linear programming problems.

2.3.2 Special Cases of Linear Programming Problems

This subsection discusses incidents that can occur when a linear programming problem has been formulated and submitted for solution. The first two cases are really error messages to the modeler. They require immediate intervention, as the solver will not be able to continue. The last three cases are of a more technical nature, it is good to know about them, but no user intervention is required. Below, we will discuss each of these issues separately and illustrate it with a numerical example.

(1) *There exists no feasible solution.*
The nonexistence of a feasible solution is related by the solver to the analyst. Whenever that happens, this must be taken as an error message. This is also the time for the user to intervene, as the solver is not able to act any further. The error message indicates that constraints are too tight, meaning that there is a contradiction among the constraints. This message is very often—but by no means exclusively—received by inexperienced analysts, who include many constraints in their model, some of which refer to situations they wish to happen rather than those that must happen. It is very important to understand that constraints are absolute, meaning that they cannot be violated.

As an illustration, consider the following numerical example. Since this special case is caused exclusively by the constraints, we will not include an objective function in our model. Suppose that the feasible set is determined by the following set of constraints:

$$
\begin{aligned}
\text{P: } & 2x_1 + 3x_2 \le 7 && (I)\\
& -x_1 + x_2 \ge 3 && (II)\\
& x_1 \ge 0 && (III)\\
& x_2 \ge 0. && (IV)
\end{aligned}
$$

A graphical representation of the problem is shown in Figure 2.3.6. Clearly, there is a contradiction between the constraints. To see this algebraically, rewrite constraint *II* as $x_2 \ge 3 + x_1$, which, as constraint *III* requires that $x_1 \ge 0$, implies that $x_2 \ge 3$. Similarly, constraint *I* can be rewritten as $3x_2 \le 7 - 2x_1$. As $x_1 \ge 0$ per constraint *III*, this implies that $3x_2 \le 7$ or, equivalently, $x_2 \le 2\frac{1}{3}$. This is an obvious contradiction to the requirement that $x_2 \ge 3$.

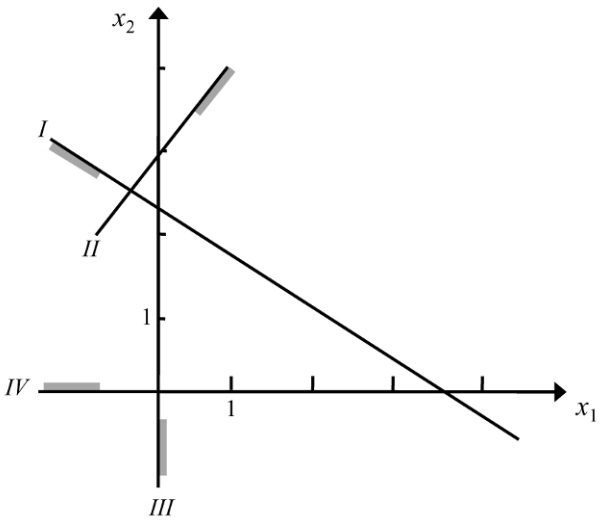


Figure 2.3.6

The question often arises as to which of the constraints actually causes the nonexistence of feasible solutions. To investigate this question, consider again the example in Figure 2.3.6. If constraint *I* were to be deleted, the feasible set would be the cone that is constructed by constraints *II* and *III* with the vertex at (0, 3). If constraint *II* were to be deleted, the feasible set is the triangle with vertices at (0, 0), (3½, 0), and (0, 2⅓). If constraint *III* were to be deleted, the feasible set would be the set that is determined by the halfspaces of constraints *I*, *II*, and *IV* which has its vertices at the points (–.4, 2.6) and (3.5, 0). Finally, if constraint *IV* were to be deleted, there would still be no feasible solution.

In summary, we have seen that the deletion of any one of the constraints *I*, *II*, and *III* causes infeasibility to disappear. This means that the question "which constraint causes the infeasibility" is indeed the wrong question: it is not a single constraint that causes infeasibility, but the incompatibility of a number of constraints, here *I*, *II*, and *III*. Many solvers will not only provide the decision maker with the "There exists no feasible solution" message, but also offer further help. Typically, this help comes in the form of an identification of the set of constraints that causes the infeasibility. If we were to add the constraints $x_1 \geq 2$ and $x_2 \geq 2$ to the region pictured in Figure 2.3.2 with constraints *I* – *VI*, there would be no feasible solution, and the solver would notify the analyst that the constraints $5x_1 + 3x_2 \leq 15$, $x_1 \geq 2$, and $x_2 \geq 2$ together cause the infeasibility.

The next question is then how to deal with infeasibilities should they occur. The answer is that the planner has to decide which of the variables involved should be "loosened up." As an example of a budget constraint, loosening up such a "≤" constraint would be accomplished by increasing the right-hand side value. In other words, allowing some additional expenditures makes the problem less stringent. Similarly, if a customer requires at least, say, 100 units, such a "≥" constraint would be made looser by reducing this number by convincing the customer to accept a somewhat smaller quantity. Even equations can be relaxed somewhat. As an example, consider the equation $2x_1 + 5x_2 = 6$, in which the left-hand side indicates the quantity of certain critical nutrients or medicines that an individual consumes. Any equation can be expresses as two opposing inequality constraints, in our case as $2x_1 + 5x_2 \leq 6$ and $2x_1 + 5x_2 \geq 6$. Relaxing these two constraints means allowing a bit less than 6 and a bit more than 6 units of that nutrient in the diet, which can be achieved by changing the right-hand side values by some small value. For instance, instead of the original constraints we could use $2x_1 + 5x_2 \leq 6.1$ and $2x_1 + 5x_2 \geq 5.9$, thus allowing a certain bandwidth within which there are feasible solutions. These are so-called *interval constraints* that marginally increase the size of the problem, but are much easier to deal with from a computational and modeling point of view.

Often, however, it is not advisable to change the right-hand side value of a single constraint, as a very significant change may be required to achieve feasibility. It is often much easier to make a number of smaller changes on multiple right-hand side values. As an example, consider the example in Figure 2.3.6. Changing the right-hand side of constraint *II* alone would require us to reduce the original value of 3 by more than 22% down to 2⅓ before a feasible solution could be obtained. Similarly, the right-hand side value of constraint I must be increased from its original value of 7 to at least 9 before a feasible solution can be obtained, that is an increase of more than 28%. Alternatively, we could increase the right-hand side value of constraint *I* by 14% to 8 and simultaneously reduce the right-hand side value of constraint *II* by 11% down to 2.67, and obtain a feasible solution with these smaller changes that may be easier to implement.

(2) *Unbounded "optimal" solutions.*
The existence of unbounded "optimal" solutions is, in some sense, the opposite of nonexisting feasible solutions. This is true in the sense that in the previous case the constraints were too tight and had to be loosened up, they are too loose here and require tightening. Most frequently, this case occurs if some constraints have been forgotten. Again, it is an error message that requires intervention from the analyst.

As an illustration of this case, consider the following numerical example.

$$
\begin{aligned}
&\text{P: Max } z = 2x_1 + x_2 \\
&\text{s.t.} \quad x_1 - x_2 \le 2 \qquad (I) \\
&\qquad -2x_1 + x_2 \le 1 \qquad (II) \\
&\qquad x_1, \quad x_2 \ge 0.
\end{aligned}
$$

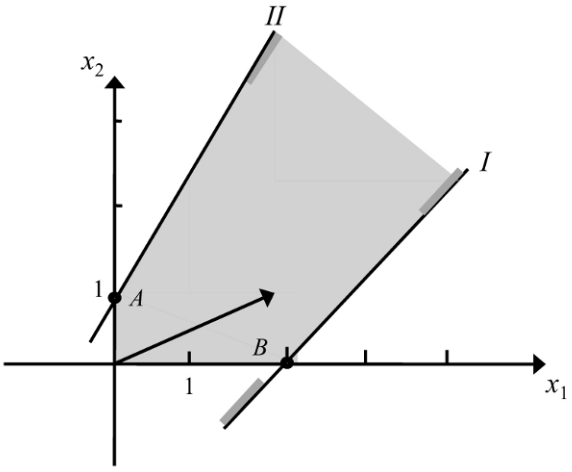


Figure 2.3.7

The graphical representation of this problem is shown in Figure 2.3.7. Using the graphical solution method, we notice that we can increase the value of the objective function to arbitrary levels by following the direction pointed out by the gradient of the objective function. (In case you should encounter arbitrarily high profits in practice, remember that you learned it from us first. We are happy with 10% of infinity). Clearly, this cannot occur in practice, which is why we use the word "optimal" in quotes. In order to resolve the issue, it helps if the analyst asks how such arbitrarily good solutions can be obtained. For instance, in a production problem we would determine that we make money by making and selling products: given that machine capacities and demand are limited, profits are limited, too. So it would benefit the modeler to investigate whether or not the appropriate constraints have been included.

At first glance, it appears that the reason for the existence of unbounded "optimal" solutions is the fact that the feasible set is not bounded in one direction (in our example, the northeast). However, if we leave the feasible set in our example unchanged, but modified the objective function from its original Max $z = 2x_1 + x_2$ to Min $z = 2x_1 + x_2$, the graphical solution method would continue to use the same contour lines, but the gradient of the new objective function would point into a southwesterly direction, diametrically opposed to its previous direction. The (finite) optimal solution is then found at the origin.

This small example illustrates that a feasible set that is unbounded in one direction is a necessary, but not sufficient, condition for the existence of unbounded "optimal" solutions. In addition, the gradient of the objective function must also point "towards that opening." Here, we will leave this imprecise wording as it is; suffice it to mention that exact mathematical conditions for the existence of unbounded "optimal" solutions do exist.

(3) *Dual degeneracy*.
This is the first of those special cases that an analyst should know about, but that does not require intervention. Formally, dual degeneracy occurs if two adjacent extreme points of the feasible set have the same value of the objective function. An example is the problem shown in Figure 2.3.5. As already discussed in the context of simplex paths, the point A has the same value of the objective function as its neighboring point F. That in itself is not really noteworthy. The situation, however, changes if dual degeneracy occurs at optimum. By definition, if we have an optimal solution and another extreme point with the same z-value exists, that point must also be optimal.

An example for this is again Figure 2.3.5, but with the original objective function changed from Max $z = 2x_1 + 3x_2$ to Min $z = 2x_1 + 3x_2$. this changes the gradient of the objective function by 180°, leaving us with the points A and F as optimal solutions. Whenever two neighboring extreme points are optimal, all points between them are also optimal, i.e., dual degeneracy at optimum means the existence of *alternative optimal solutions*. In this example, the optimal coordinates of the points A and F are (0, 1) and (1½, 0), respectively, both having an objective value of $z = 3$. Some points on the line segment between these two points are $\bar{\mathbf{x}}$ = (¾, ½), ($\frac{9}{8}$, $\frac{1}{4}$), and ($\frac{3}{8}$, $\frac{3}{4}$). These points have an objective value of $z = 3$ and thus are optimal as well. However, none of them is an extreme point, and the simplex method will not generate them.

Note that the fact that points that are not corner points of the feasible set are optimal does not invalidate Dantzig's corner point theorem. All the theorem states is that *at least* one optimal solution is at an extreme point. This allows for the possibility of nonextreme points being optimal, but only if there is another optimal extreme point as well.

(4) *Redundancy*.
Redundancy is an issue that relates to individual constraints. In particular, a constraint is said to be redundant, if it can be deleted from the set of constraints without changing the feasible set. A more detailed definition will distinguish between constraints that exhibit *strong redundancy* and those that are *weakly redundant*. While a weakly redundant constraint belongs to a constraint whose hyperplane shares at least one point with the feasible set, this is not the case for strongly redundant constraints.

As an illustration, consider the following numerical example (without an objective function, as it does not play a role in the discussion of redundancy).

$$
\begin{aligned}
\text{P: } & 3x_1 + x_2 \le 8 && (I)\\
& x_1 \le 2 && (II)\\
& x_1 - x_2 \le 3 && (III)\\
& x_1 + 2x_2 \le 6 && (IV)\\
& x_1, \quad x_2 \ge 0.
\end{aligned}
$$

Figure 2.3.8 depicts the feasible set of this problem.

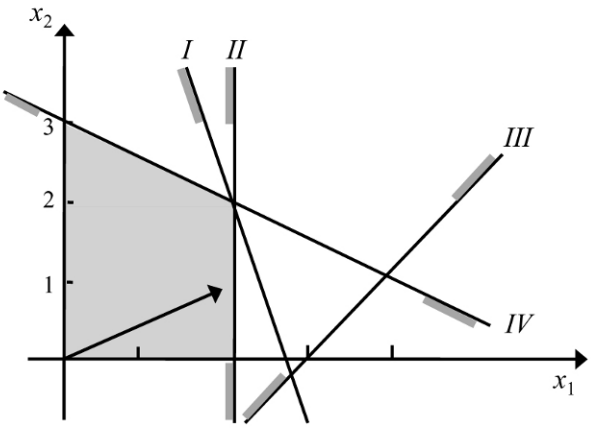


Figure 2.3.8

Constraint *III* is obviously redundant as its removal does not change the shaded area that symbolizes the feasible set. As a matter of fact, it is strongly redundant as its bordering hyperplane does not touch the feasible set at any point. In contrast, consider constraint *I*. It is also redundant (but just about so), so that its removal likewise does not change the feasible set. However, it is only weakly redundant, as it touches the feasible set at the point (2, 2). Had constraint *I* be instead $3x_1 + x_2 \le 8.1$, it would have been strongly redundant, had it been $3x_1 + x_2 \le 7.95$, it would not have been redundant at all, as it would have cut off the point (2, 2) and thus shaped the feasible set.

By their very definition, redundant constraints can be deleted from the problem formulation without changing anything. The problem is that there is no easy general way to recognize whether or not a constraint is redundant. Again, the graphical representation is deceiving as in it, redundancy can easily be detected. However, if all we have is a set of constraints that we cannot plot, detection of redundancy is a different, and much more complicated, matter. As a matter of fact, determining whether or not a constraint is redundant is as difficult as solving the existing linear programming in the first place. And this is why we can, and pretty much have to, ignore the issue and have the solver compute optimal solutions, regardless whether or not redundant constraints exist in the formulation.

(5) *(Primal) degeneracy*.
While the issue of primal degeneracy is of very little, if any, practical concern, it is a very important theoretical matter. If primal degeneracy is not properly taken care of, the simplex method may "cycle," meaning that it generates the same point over and over again without ever reaching an optimal solution. In graphical terms, primal degeneracy occurs in two dimensions, if more than two planes of constraint intersect at a single point. In Figure 2.3.8, the point (2, 2) exhibits degeneracy. As in two dimensions, the intersection of any two straight lines uniquely determines a point, we can think of points with primal degeneracy as "overdetermined." In n dimensions, primal degeneracy occurs if the hyperplanes of more than n constraints intersect at one point. Any modern code will include a module that deals with degeneracy, so that this is of no concern to users.

Exercises

Problem 1 (graphing constraints and objective, graphical solution method): Consider the following linear programming problem.

$$
\begin{array}{lll}
\text{Max } z = x_1 + x_2 & & \\
\text{s.t.} \quad 5x_1 + 2x_2 \le 10 & & (I) \\
\qquad\quad x_2 \ge 1 & & (II) \\
\qquad 3x_1 + 5x_2 \le 15 & & (III) \\
\qquad x_1, \quad x_2 \ge 0. & &
\end{array}
$$

(a) Graph the constraints, determine the feasible set, and use the graphical solution method to determine the optimal solution point. Which constraints are binding at optimum? Compute the exact coordinates at optimum and calculate the value of the objective function at optimum.
(b) What if the objective were Min $z = -3x_1 - x_2$? Plot the new objective, use the graphical solution method, determine the optimal solution point, its coordinates, and the value of the objective function at optimum.
(c) Repeat (b) with the objective function Min $z = x_1 - 2x_2$

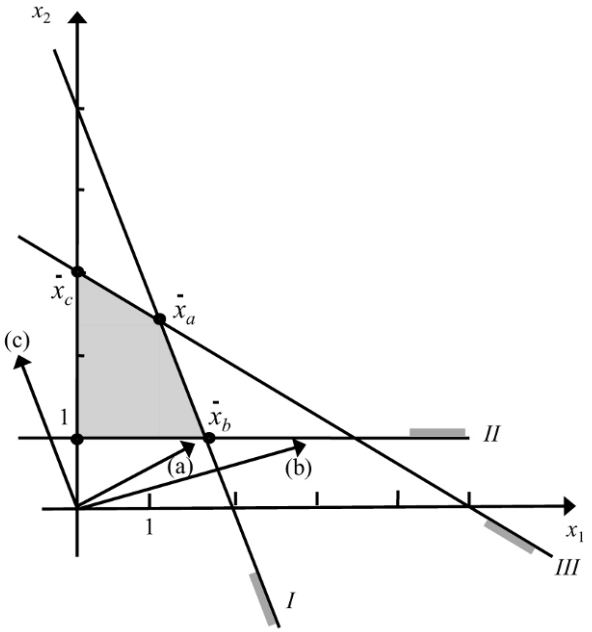


Figure 2.3.9

Solution:
The objective functions for (a), (b), and (c) are shown in Figure 2.3.9. The optimal solutions are indicated by the points $\bar{\mathbf{x}}_a, \bar{\mathbf{x}}_b$, and $\bar{\mathbf{x}}_c$, respectively. The exact coordinates are $\bar{\mathbf{x}}_a = [1.0526, 2.3684]$ with $\bar{z}_a = 3.4211$, $\bar{\mathbf{x}}_b = [1.6, 1]$ with $\bar{z}_b = -5.8$, and $\bar{\mathbf{x}}_c = [0, 3]$ with $\bar{z}_c = -6$. The binding constraints at the three points are *I* and *III*, *I* and *II*, and *III* and the nonnegativity constraint $x_1 \geq 0$, respectively.

Problem 2 (graphing constraints and objective, graphical solution method): Consider the following linear programming problem.

$$
\begin{array}{lll}
\text{Max } z = 3x_1 + x_2 & & \\
\text{s.t.} \quad 6x_1 + 5x_2 \geq 30 & & (I) \\
\quad -x_1 + 2x_2 \geq 4 & & (II) \\
\quad x_2 \leq 5 & & (III) \\
\quad x_1, \quad x_2 \geq 0. & &
\end{array}
$$

(a) Graph the constraints, determine the feasible set, and use the graphical solution method to determine the optimal solution point. Which constraints are binding at optimum? Compute the exact coordinates at optimum and calculate the value of the objective function at optimum.

(b) Repeat (a) with the objective function Min $z = x_2$.
(c) What happens if constraint *I* were of the type "≤" instead?

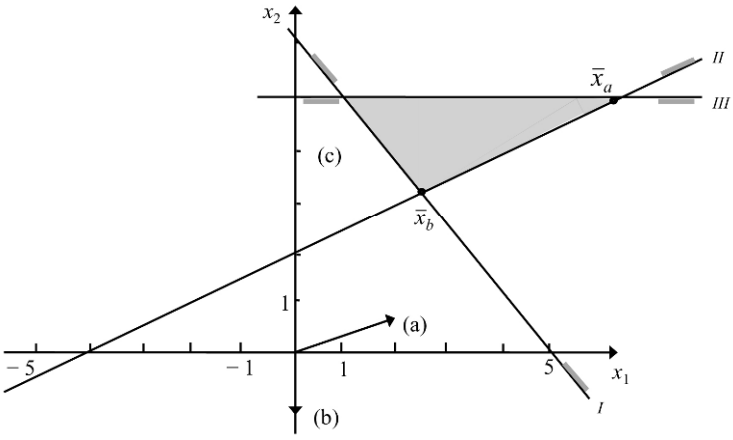


Figure 2.3.10

Solution:

(a) The feasible set is the shaded area in Figure 2.3.10, and the optimal solution point is $\bar{\mathbf{x}}_a$ = [6, 5] with $\bar{z}_a$ = 23. Constraints *II* and *III* are binding at optimum.
(b) The objective leads "straight down" as shown in the above figure. The optimal solution point is $\bar{\mathbf{x}}_b$ = [2.3529, 3.1765] with the objective value $\bar{z}_b$ = 3.1765. At this point, the constraints *I* and *II* are binding.
(c) The feasible set would no longer be the shaded region but the quadrilateral indicated by (c). Given the objective in (a), the optimal solution is again $\bar{\mathbf{x}}_b$.

2.4 Postoptimality Analyses

This section will investigate what can be thought of as the third phase in linear programming. In particular, it asks: "What happens, if…?" In simple words, we will examine what happens, if there is some change in some of the components of the linear programming problem that was formulated and solved earlier. The first subsection will explore what it means graphically when we make the proposed change, while the second subsection puts this knowledge to work and examines what managerial consequences the anticipated changes will have.

2.4.1 Graphical Sensitivity Analyses

Recall that one of the main assumptions of linear programming is the deterministic property. In other words, we assume that the structure of the problem as well as all

of the parameters of the problem are assumed to be known with certainty. In virtually all realistic cases, this is a troubling assumption: prices may or may not be known in advance, demand may be uncertain, machine capacities may change (due to unforeseen breakdowns), employees may call in sick, thus changing the availability of manpower, and so forth. How then can we justify using linear programming at all?

Using *postoptimality analyses* can be seen as a "trick" that allows us to get away with assuming that the deterministic property holds, while it actually does not. As an example, suppose that we are facing an uncertain demand, which, as past experience indicates, ranges from, say, 80 to 130 units. Furthermore, suppose that most of the time the demand is somewhere about 110 units. The idea is now to temporarily fix the demand at the level of 110 and solve the problem. Once this has been done, we perform sensitivity analyses by asking: "What happens, if the demand were to increase (from 110) by 10 (to 120)?" "What if it increases by 20 (to 130)?" "What if it decreases by 10, 20, or 30 units?" Information of this type can be obtained either by setting the demand to 120, 130, or any of the other values the decision maker is interested in and actually resolving the problem, or by gleaning the information from the printout. This information will then indicate how sensitive the solution is to changes in the input parameters. While some problems are not sensitive at all, meaning that even significant changes in the original demand, prices, or other parameters do not result in major (or even any) changes of the solution, others are very sensitive, so that even minor changes in the input parameters change the solution a great deal. There is nothing we can do about it, but it is very important information to the decision maker. If it is known that a model is very sensitive to changes, the decision maker will have to be very cautious and monitor the process closely, obtaining additional and updated information at many steps along the way. This is not as necessary in problems that are rather insensitive to changes.

We distinguish between two different types of sensitivity analyses. The first type deals with *structural changes*, meaning the addition and deletion of variables and constraints. Changes of that nature are major and often dealt with by re-solving the problem altogether. The second type of sensitivity analyses involves *parameter changes*. In other words, in these cases numbers of the model change. Typically, we deal with *ceteris paribus* changes, i.e., we determine what happens if one number changes, while all other parameters remain unchanged. The advantage of such an analysis is that it separates the different changes and analyzes their effect. If we were to analyze simultaneous changes of a number of parameters, we would not be able to specify what actually causes, say, the increase or decrease in the total costs.

We will first look at the changes that may occur when we either add or delete a variable or a constraint. The addition of variables and constraints is an issue during the modeling process when a model is built. Furthermore, it may occur after the problem has been solved when opportunities and requirements are added

to the problem as time passes. Similarly, it may happen that over time some activities or possibilities no longer exist or constraints have become obsolete. Whenever possible, we will deal with these cases on an intuitive level, which is pretty much all that can be done short of re-solving the problem.

Consider the *addition of a variable*. In graphical terms, the addition of a variable corresponds to the increase of the dimensionality of the problem. It is useful to understand a variable as an opportunity, i.e., an additional activity that we may or may not undertake. Having the variable allows us to choose the level at which we engage in the activity (i.e., the value the variable assumes), while not including the variable in the model is the same as setting its activity level or value equal to zero. In this sense, adding a variable is the same as adding an opportunity. Doing so allows us to increase the level of the new possible activity from a zero level to any value within the constraints. The immediate conclusion is that the addition of a variable can never result in a deterioration of the objective function value (a decrease in maximization functions or an increase in minimization functions), but possibly an improvement. Formally, defining the original problem as P_{orig} with its optimal objective value $\bar{z}_{orig}$ and the problem with the added variables as P_{addvar} and its optimal objective value as $\bar{z}_{addvar}$, we know that $\bar{z}_{orig} \leq \bar{z}_{addvar}$ in problems with maximization objective and $\bar{z}_{orig} \leq \bar{z}_{addvar}$ in problems with minimization objective. Along similar lines, if P_{orig} has unbounded "optimal" solutions, then so does the modified problem P_{addvar}. On the other hand, if the original problem P_{orig} has no feasible solution, the added variable may (or may not) allow P_{addvar} to have feasible solutions.

There is little than can be said beyond this. Often, if variables are added in the modeling process, it may be useful not to start solving the new problem from scratch but to start with the previous optimal solution, which usually only requires a few additional steps to reach the optimal solution of the new problem. However, with the advances of optimization software, it often takes only seconds to re-solve a problem, so that such considerations are no longer required.

The *deletion of a variable* can be discussed in analogous fashion. Again, let the original problem be P_{orig}, while the formulation without the now deleted variable is P_{delvar}. The objective function values of the two problems are defined as $\bar{z}_{orig}$ and $\bar{z}_{delvar}$, respectively. Deleting a variable is now equivalent to deleting an opportunity, or, more formally, forcing the value of a variable to zero. If the value of the variable that is to be deleted equaled zero in P_{orig}, then the variable can be deleted without any change of the solution and $\bar{z}_{orig} = \bar{z}_{delvar}$ in both, maximization and minimization problems. The main reason for this result is that we are no longer allowing an activity that we did not engage in in the first place. On the other hand, if the value of the variable that is to be deleted was positive in the original problem P_{orig}, the deletion of this variable will deprive us from a

worthwhile activity, which lets the objective value deteriorate. In other words, we have $\bar{z}_{orig} \geq \bar{z}_{delvar}$ for maximization and $\bar{z}_{orig} \leq \bar{z}_{delvar}$ for minimization problems. It is also straightforward that if P_{delvar} has unbounded "optimal" solutions, then so does P_{orig}, and if P_{orig} has no feasible solution, then neither does P_{delvar}.

The *addition of a constraint* is an issue that we can much more easily analyze, as it allows us to visualize the situation in a graph. Again, we restrict ourselves to small problems with only two variables, but the conclusions are valid for any general formulation. Consider again some arbitrary original maximization problem P_{orig} and assume that a constraint is added, resulting in P_{addcon}. By definition, adding a constraint means that the resulting problem P_{addcon} is more restricted, so that its objective value $\bar{z}_{addcon} \leq \bar{z}_{orig}$ in maximization problems and $\bar{z}_{addcon} \geq \bar{z}_{orig}$ in minimization problems. More specifically, if the new constraint is satisfied by the optimal solution of P_{orig}, then this solution will also be optimal for the new problem P_{addcon}. (Note that this is the case if the new constraint is either redundant or essential, but not binding at optimum). On the other hand, if the old optimal solution violates the new constraint, then the optimal solution of P_{addcon} is different from the optimal solution of P_{orig} and the objective value will be the same (in case of alternative optimal solutions) or be worse than before. Furthermore, if the original problem has unbounded "optimal" solutions, then the problem with the new constraint may or may not have bounded optimal solutions. If the original problem has no feasible solutions, then adding a variable and making it more constrained will not result in feasible solutions.

Finally, consider the *deletion of a constraint*. Again, assume that the problem P_{orig} has been solved, resulting in the objective value $\bar{z}_{orig}$. The problem without one or more of the constraints will be referred to as P_{delcon} and its optimal value of the objective function is $\bar{z}_{delcon}$. It is apparent that the new problem P_{delcon} is less restricted than the original problem P_{orig}, so that $\bar{z}_{delcon} \geq \bar{z}_{orig}$ holds for maximization problems, while $\bar{z}_{delcon} \leq \bar{z}_{orig}$ holds for any minimization problem. As in the case of constraint additions, we can distinguish between two cases: either the constraint that is deleted was binding at optimum before it was deleted, or it was not. In case it was binding, then it was, generally speaking, a constraint that held back the solution and with its removal, better solutions may exist. On the other hand, if the constraint was not binding, then it did not hold back in the solution in the original problem, so that its removal cannot result in better solutions. Again, if unbounded "optimal" solutions existed in P_{orig}, then the removal of a constraint cannot change that regardless if it is binding or not. If P_{orig} did not have feasible solutions, the deletion of a constraint may or may not result in the problem P_{delcon} having feasible solutions.

Next consider parameter changes. In order to classify such changes, consider the following simple linear programming problem

$$
\begin{aligned}
\text{P: Max } z &= 5x_1 + 6x_2 \\
\text{s.t.} \quad & x_1 - 2x_2 \geq 2 \\
& 3x_1 + 4x_2 \leq 12 \\
& x_1, \quad x_2 \geq 0.
\end{aligned}
$$

This model, as well as any other linear programming problem, includes three different types of parameters. The first are the *objective function coefficients* (typically denoted by c_1, c_2, …), which are the numbers found in the objective function (here the numbers $c_1 = 5$ and $c_2 = 6$). Depending on the application, they may be referred to as unit profits, cost coefficients, or similar names. The second type of parameter are the *right-hand side values* (which we usually denote by b_1, b_2, …). In our example, these are the values $b_1 = 2$ and $b_2 = 12$. Again, depending on the specific applications, the right-hand side values may be referred to as resource availabilities, demands, inventory levels, or similar names. Finally, there are the *left-hand side coefficients*, which sometimes are called *technological coefficients* a_{11}, a_{12}, …, a_{21}, a_{22}, … with the first subscript denoting the number of the row or constraint and the second subscript standing for the number of the column or variable. In our example the technological coefficients are $a_{11} = 1$, $a_{12} = -2$, $a_{21} = 3$, and $a_{22} = 4$. Depending on the application, these values may symbolize the processing times of a product on a machine, the content of nutrients of a food item, the interest rate of a specific investment, or similar values. In this book, we will investigate changes of the objective function coefficients and changes of the right-hand side values. For changes of the left-hand side parameters, we suggest to simply re-solve the problem.

First, consider changes of the objective function coefficients. In order to explore what these parameter changes cause, we first look at the objective function and ignore the constraints. To facilitate our arguments, consider the objective function Max $z = 3x_1 + 2x_2$. The gradient of the objective is shown in Figure 2.4.1 and is labeled (3, 2). Suppose now that we want to examine changes of c_1, the value associated with the variable x_1. If this number, whose original value is "3," decreases to, say, "2," the gradient tilts in a counterclockwise direction to a position shown as (2, 2). If, on the other hand, c_1 were to increase to, say, "4," then the gradient will tilt in a clockwise direction to the position shown as (4, 2). If c_1 were to further increase to a value of "5," the gradient further tilts in a clockwise direction to the position shown as (5, 2).

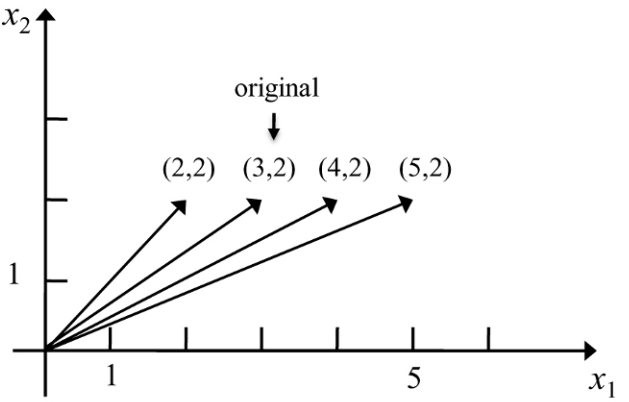


Figure 2.4.1

We see that the increase of an objective function coefficient in a maximization function symbolizes the fact that the activity that corresponds to its associated variable has become more attractive or profitable. Thus the gradient of the objective function is drawn more into that direction. In our example, we see that as c_1 increases from 3 to 4 and then to 5, the gradient is pulled further and further into the x_1 direction, i.e., to the right.

We note that since all the changes occur in the objective function, the feasible set will remain unaffected by these changes. This means that if there are no feasible solutions before the change, then there will be no feasible solutions after the change. On the other hand, the case of unbounded "optimal" solutions is different, as it does depend not only on the feasible set, but also on the gradient of the objective function. As a simple example, consider the following linear programming problem.

$$\begin{aligned} \text{P: Max } & 2x_1 + 1x_2 \\ \text{s.t.} \quad & x_1 - x_2 \leq 2 \\ & x_1, \quad x_2 \geq 0. \end{aligned}$$

This problem is shown in Figure 2.4.2, in which the gradient of the objective function is labeled by its coefficient as (2, 1). Clearly, there are unbounded "optimal" solutions to the problem. However, if c_2 decreases from its present value of "1" to "–2" or even less (the gradient is shown as (2, –2)), the problem has a unique finite optimal solution at $\bar{x}_1 = 2$ and $\bar{x}_2 = 0$ with an objective function value of $\bar{z} = 4$, clearly a finite value.

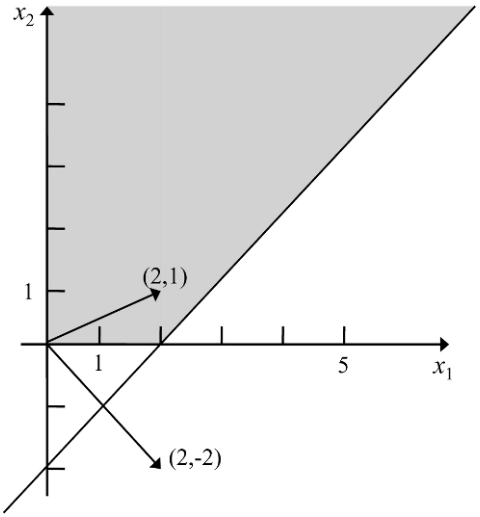


Figure 2.4.2

We are now able to incorporate constraints in our discussion. In order to do so, consider the following linear programming problem:

$$
\begin{aligned}
&\text{P: Max } z = 1x_1 + 2x_2 \\
&\text{s.t.} \qquad\quad x_2 \le 3 \\
&\qquad\quad 3x_1 + 2x_2 \le 11 \\
&\qquad\quad x_1 - x_2 \le 2 \\
&\qquad\quad x_1, \quad x_2 \ge 0.
\end{aligned}
$$

The problem can be visualized in the graph in Figure 2.4.3.

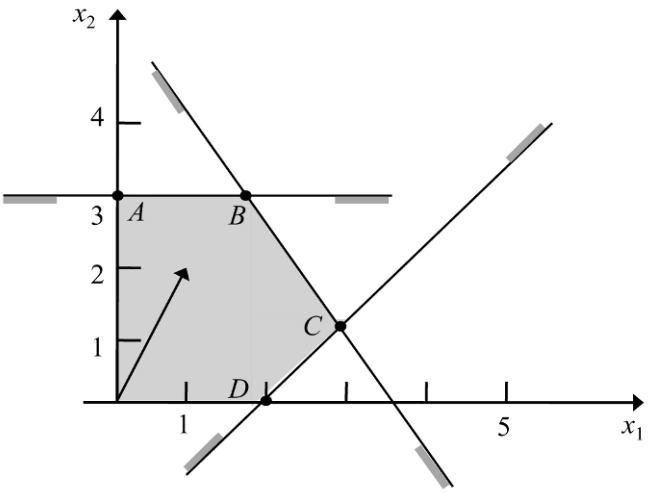


Figure 2.4.3

Using the graphical solution technique (or any solver), we determine that the point B is the unique optimal solution with coordinates $(x_1, x_2) = (1⅔, 3)$. Suppose now that we want to examine the sensitivity of the solution with respect to c_2, the objective function coefficient of x_2. If we were to increase the value that presently equals 2 to some higher value, our previous discussion indicates that the gradient of the objective function tilts in a counterclockwise direction. This does not, however, have any effect on the solution, which stays at point B. As a matter of fact, no finite value of c_2, regardless of how large, will change the solution, which remains at point B (meaning that the solution is very insensitive to increases of this coefficient). Clearly, since $\bar{x}_2 = 3$ at optimum, the value of the objective function will change as the activity that the variable x_2 symbolizes becomes more and more valuable.

Consider now a decrease of c_2. Graphically, this means that the gradient of the objective function tilts in a clockwise direction. For small changes, the optimal solution remains at point B. However, once c_2 reaches the value of ⅔, point B is still optimal, but so is point C with coordinates $(x_1, x_2) = (3, 1)$ (and all non-extreme points on the line segment between these two points). This is clearly a case of dual degeneracy at optimum, i.e., alternative optimal solutions. Once c_2 decreases below the value of ⅔, point C is the unique optimal solution. Point C remains optimal c_2 reaches the value of -1. At this point, points C and D are both optimal, again a case of alternative optimal solutions. If c_2 drops below -1, point D with coordinates $(x_1, x_2) = (2, 0)$ remains optimal, regardless how small the coefficient is.

We summarize the effects of the changes c_2 in Table 2.4. 1.

Table 2.4.1: Optimal solutions $(\bar{x}_1, \bar{x}_2)$ and objective values $\bar{z}$ for different values of c_2

Range of c_2	$]-\infty, -1[$	-1	$]-1, \frac{2}{3}[$	$\frac{2}{3}$	$]\frac{2}{3}, +\infty[$
Optimal solution point	D	D and C	C	C and B	B
Optimal coordinates $(\bar{x}_1, \bar{x}_2)$	(2, 0)	(2, 0) and (3, 1)	(3, 1)	(3, 1) and (1⅔, 3)	(1⅔, 3)
Optimal objective value $\bar{z}$	2	2	$3 + c_2$	3⅔	$1\frac{2}{3} + 3c_2$

Similar analyses can be performed for each of the given objective function coefficients. Without further comments, Table 2.4.2 shows different ranges of c_1 and the resulting optimal solution points, their coordinates, and their values of the objective function.

Table 2.4.2: Optimal solutions $(\bar{x}_1, \bar{x}_2)$ and objective values $\bar{z}$ for different values of c_1

Range of c_1	$]-\infty, 0[$	0	$]0, 3[$	3	$]3, \infty[$
Optimal solution point	A	A and B	B	B and C	C
Optimal coordinates $(\bar{x}_1, \bar{x}_2)$	(0, 3)	(0, 3) and (1⅔, 3)	(1⅔, 3)	(1⅔, 3) and (3, 1)	(3, 1)
Optimal objective value $\bar{z}$	6	6	$6 + 1\frac{2}{3}c_1$	11	$2 + 3c_1$

So far we have only looked at the effects of individual changes on the optimal solution. There is, however, an interesting result that considers the effects of simultaneous changes. The rule is called the *100 percent rule* and it can be stated as follows.

100% Rule: As long as the sum of the absolute values of the increases or decreases of the objective function coefficients is no more than 100%, the optimal solution point remains optimal.

Formally, we denote the largest allowable increase of an objective function coefficient c_j by $\overline{\Delta} c_j$, while the largest allowable decrease of an objective function coefficient c_j is denoted by $\underline{\Delta} c_j$. In our example, the optimal solution for the original objective function Max $z = 1x_1 + 2x_2$ was $(\bar{x}_1, \bar{x}_2) = (1\frac{2}{3}, 3)$. As shown in Table 2.4.2, this solution remains optimal as long as c_1 (whose original value is

$c_1 = 1$) does not increase by more than $\overline{\Delta}c_1 = 2$ to the upper limit of the range at $c_1 = 3$. Similarly, the solution remains optimal as long as c_1 does not decrease by more than $\underline{\Delta}c_1 = 1$ to the lower end of the range at $c_1 = 0$. Similarly, we obtain the values $\overline{\Delta}c_2 = +\infty$ and $\underline{\Delta}c_2 = 1\frac{1}{3}$. Suppose now that we want to investigate the effect of a simultaneous increase of the value of c_1 by Δc_1 and a decrease of the value of c_2 by Δc_2, the 100 percent rule then states that the optimal solution $(\bar{x}_1, \bar{x}_2) = (1\frac{2}{3}, 3)$ remains optimal, as long as the sum of actual increases in relation to their respective values $\overline{\Delta}c_j$ plus the sum of actual decreases in relation to their respective values $\underline{\Delta}c_j$ does not exceed 100% = 1. In our example, we obtain $\frac{|\Delta c_1|}{\overline{\Delta}c_1} + \frac{|\Delta c_2|}{\underline{\Delta}c_2} = \frac{|\Delta c_1|}{2} + \frac{|\Delta c_2|}{1\frac{1}{3}} \leq 1$. For instance, if we were to face a simultaneous increase of c_1 by ½ and a decrease of c_2 by ½, the condition tells us that $\frac{½}{2} + \frac{½}{1\frac{1}{3}} = ¼ + ⅜ = \frac{5}{8} < 1$, so that the optimal solution remains optimal. On the other hand, if the increase of c_1 were ¾ and the decrease of c_2 were 1, then we would have $\frac{3/4}{2} + \frac{1}{1\frac{1}{3}} = ⅜ + ¾ = \frac{9}{8} > 1$, so that the optimal solution will change.

A similar argument can be applied to a simultaneous decrease of c_1 and increase of c_2, or simultaneous increases or decreases of both cost coefficients. Increasing both cost coefficients simultaneously presents an interesting special case. As there is no finite upper bound on c_2, the 100% rule reduces to the regular limit on c_1 and no limit on c_2.

Consider now changes of a single right-hand side value b_i. Again, we will first examine the effects such a change on the constraint itself before investigating what happens to optimal solutions. As a numerical example, consider the constraint $2x_1 + 3x_2 \leq 6$. The resulting hyperplane and halfspace is shown in Figure 2.4.4, labeled as $b_1 = 6$.

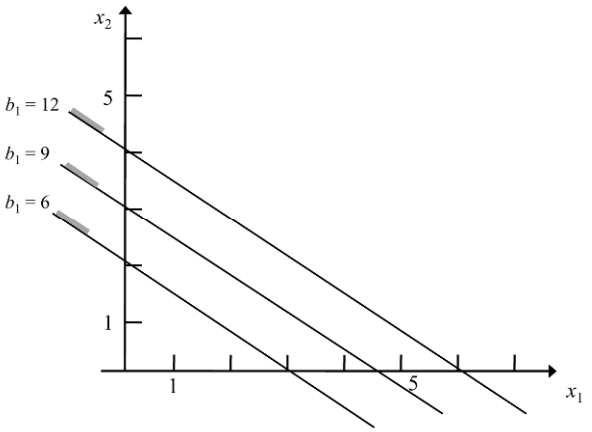


Figure 2.4.4

If we were to modify the right-hand side value to, say, $b_1 = 9$, the hyperplane and halfspace would shift in parallel fashion to the position shown in Figure 2.4.4 by $b_1 = 9$. A further increase to $b_1 = 12$ is shown in the figure as well. Decreases of the right-hand side value in our example would again result in a hyperplane and/or halfspace shifting in parallel fashion, but now in a southwesterly direction.

Given this result, we are now able to examine changes of right-hand side values in a linear programming problem. Before doing so, we note that such changes do in no way affect the objective function, but they may change the feasible set.

As an illustration, consider again the numerical example that was used to discuss changes of the objective function. For convenience, we restate the model here.

$$
\begin{array}{lrll}
\text{P: Max } z = 1x_1 + 2x_2 & & & \\
\text{s.t.} & x_2 \leq 3 & & (I) \\
& 3x_1 + 2x_2 \leq 11 & & (II) \\
& x_1 - x_2 \leq 2 & & (III) \\
& x_1, \quad x_2 \geq 0. & &
\end{array}
$$

Figure 2.4.5 shows the feasible set (the shaded area) and the gradient of the objective function. The extreme points of the feasible set are *0*, *A*, *B*, *C*, and *D*. The optimal solution is again at the point *B* with coordinates $(\bar{x}_1, \bar{x}_2) = (1⅔, 3)$ and value of the objective function $\bar{z} = 7⅔$.

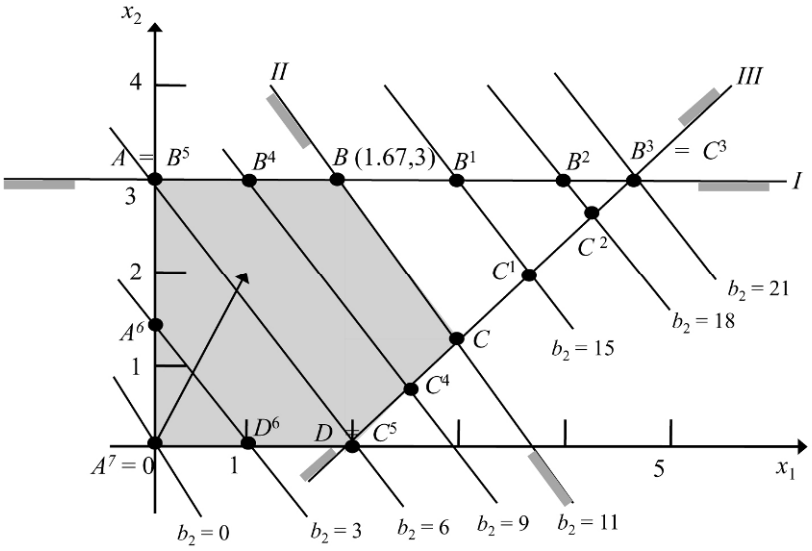


Figure 2.4.5

Consider now changes of the second right-hand side value b_2. If b_2 increases, then the hyperplane and halfspace of this constraint will move into a northeasterly direction. For instance, if $b_2 = 15$, the feasible set is now enlarged by the set with the extreme points B, C, B^1, and C^1. This has a direct effect on the optimal solution point, which has moved from its original position at point B to its new position at point B^1. Note that while the optimal coordinates of the optimal point have changed, one thing has not: the point is still determined by the intersection of the hyperplanes that belong to constraints *I* and *II*, the latter now with its new right-hand side value. Such a point is referred to as a *basis point* or simply a *basis*. We observe here that while the basis has not changed, the optimal solution has moved to a new location. A further increase of b_2 has the effect of changing the feasible set to *0*, A, B^2, C^2, and D, with an optimal solution at point B^2. Again, the basis has not changed (the point is still determined as the intersection of hyperplanes *I* and *II*), but its location has moved further to the right. A further increase to $b_2 = 21$ results in a feasible set with extreme points *0*, A, $B^3 = C^3$, and D. The optimal solution is now at points $B^3 = C^3$. We observe that at this point, hyperplanes *I*, *II*, and *III* now intersect at the same point, causing primal degeneracy. Any further increase of b_2 will not change the feasible set any further, constraint *II* is now redundant.

Return now to the original value of $b_2 = 11$ and slowly decrease this value. For $b_2 = 9$, for instance, the feasible set has shrunk to *0*, A, B^4, C^4, and D with an optimal solution at B^4. A further decrease to $b_2 = 6$ results in the feasible set with extreme points *0*, $A = B^5$, and $D = C^5$, indicating that primal degeneracy now occurs at $A = B^5$, and $D = C^5$. The optimal solution is now at $A = B^5$. A further decrease to $b_2 = 3$

results in a feasible set with extreme points 0, A^6, and D^6 with an optimal solution at A^6. A further decrease to $b_2 = 0$ causes the feasible set to consist of only the origin, which, as it is the only feasible point, is now also optimal. Any further decrease of b_2 will cause the problem not to have any feasible solutions. Our results are summarized in Table 2.4.4.

In general, our discussion has revealed that changing a right-hand side value from $-\infty$ to $+\infty$ (or vice versa) causes the constraint to be redundant at first, then essential (shaping the feasible set, but possibly not changing the optimal solution), binding (shaping the feasible set and any change of the right-hand side value, regardless how small, changes the optimal solution), to so strong so as to cause the nonexistence of feasible solutions. Not each case has to go through all of these phases. For instance, changing the value of b_2 will result in the constraint first being redundant, then essential and binding, and then infeasible.

The ranges for changes of b_1 and b_3 are shown in Table 2.4.3 and Table 2.4.4, respectively. We must note, though, that the interpretation of these intervals is different from that in the case of changes in the objective function. While a range of c_j of, say, [3, 7] indicates that as long as c_j is anywhere between 3 and 7, the optimal solution will not change, the same interval for b_i will indicate that as long as b_i is within this range, the optimal basis will not change. In other words, for all changes within this range, the same constraints will be binding at optimum. The solution will, however change. This makes the ranges for right-hand side values less interesting to managers.

Table 2.4.3: Optimal solution and objective values of sample problem of various values of b_1

Range of b_1	$]-\infty, 0[$	0	$]0, 1[$	1	$]1, 5\frac{1}{2}[$	$5\frac{1}{2}$	$]5\frac{1}{2}, \infty[$
Optimal coordinates $(\bar{x}_1, \bar{x}_2)$	There exists no feasible solution	(2, 0)	$(2+b_1, b_1)$	(3, 1)	$(-\frac{2}{3}b_1+3\frac{2}{3}, b_1)$	(0, 5½)	(0, 5½)
Optimal objective value $\bar{z}$		2	$2+3b_1$	5	$7\frac{1}{3}-\frac{1}{3}b_1$	11	11

Table 2.4.4: Optimal solution and objective values of sample problem of various values of b_2

Range of b_2	$]-\infty, 0[$	0	$]0, 6[$	6	$]6, 21[$	21	$]21, \infty[$
Optimal solution point	There exists no feasible solution	*0*	$A^7, \ldots, A$	$A = B^5$	$B^5, B^4, B, B^1, B^2, B^3$	$B^3 = C^3$	$B^3 = C^3$
Optimal coordinates $(\bar{x}_1, \bar{x}_2)$		(0, 0)	$(0, \frac{1}{2}b_2)$	(0, 3)	$(\frac{1}{3}b_2-2, 3)$	(5, 3)	(5, 3)
Optimal objective value $\bar{z}$		0	b_2	6	$4+\frac{1}{3}b_2$	11	11

Table 2.4.5: Optimal solution and objective values of sample problem of various values of b_3

Range of b_3	$]-\infty, 3[$	-3	$]-3, -1\frac{2}{3}[$	$-1\frac{2}{3}$	$]-1\frac{2}{3}, \infty[$
Optimal coordinates $(\bar{x}_1, \bar{x}_2)$	There exists no feasible solution	(0, 3)	$(3+b_3, 3)$	$(1\frac{2}{3}, 3)$	$(1\frac{2}{3}, 3)$
Optimal objective value $\bar{z}$		6	$9+b_3$	$7\frac{2}{3}$	$7\frac{2}{3}$

Finally in this section, we want to apply the 100% rule to right-hand side changes. The rule itself is the same as that used for changes of objective function coefficients. Here, with right-hand side values of $b_1 = 3$, $b_2 = 11$, and $b_3 = 2$, we have obtained intervals of [1, 5½], [6, 21], and [−1⅓, ∞[, see Tables 2.4.3, 2.4.4, and 2.4.5. In other words, we have $\underline{\Delta}b_1 = 2, \overline{\Delta}b_1 = 2\frac{1}{2}$, $\underline{\Delta}b_2 = 5, \overline{\Delta}b_2 = 10$, $\underline{\Delta}b_3 = 3\frac{1}{3}$, and $\overline{\Delta}b_3 = \infty$. With anticipated changes of the right-hand side values from their present values to $b_1 = 4\frac{1}{2}$, $b_2 = 7$, and b_3 unchanged at 2, we obtain $\Delta b_1 = 1\frac{1}{2}$, $\Delta b_2 = -4$, and $\Delta b_3 = 0$. We can then compute $\frac{|\Delta b_1|}{\overline{\Delta}b_1} + \frac{|\Delta b_2|}{\underline{\Delta}b_2} = \frac{1\frac{1}{2}}{2\frac{1}{2}} + \frac{4}{5} = \frac{7}{5} > 1$, so that these changes will not only cause the solution to change, but also the basis, i.e., different constraints will be binding after the change.

As a different example, assume that b_1 decreases from its present value of 3 to 2½ (meaning that $\Delta b_1 = -\frac{1}{2}$), b_2 increases by $\Delta b_2 = 2$ to its new value of $b_2 = 13$, and the third right-hand side value decreases by $\Delta b_3 = -1$ from its present value of 2 to its new value of $b_3 = 1$. The condition is then $\frac{|\Delta b_1|}{\underline{\Delta}b_1} + \frac{|\Delta b_2|}{\overline{\Delta}b_2} + \frac{|\Delta b_3|}{\underline{\Delta}b_3} = \frac{\frac{1}{2}}{2} + \frac{2}{10} + \frac{1}{10/3} = \frac{3}{4} < 1$, so that after this change, the same constraints will be binding. Again, the solution may (and most likely will) change.

2.4.2 Economic Analysis of an Optimal Solution

In this section, we will first provide a simulated printout that is typical for what analysts obtain from a computer upon solving a linear programming problem. We then explain the different features and the information provided by the printout. In the second part of this section, we use a linear programming problem, provide the printout, and answer a number of questions relevant to the decision maker.

In order to do so, we consider the following linear programming problem:

$$
\begin{aligned}
\text{P: Max } z = {} & 3x_1 - x_2 \\
\text{s.t.} \quad & x_1 - x_2 = 2 \\
& 2x_1 + 3x_2 \leq 16 \\
& 5x_1 + x_2 \geq 15 \\
& x_1, \quad x_2 \geq 0.
\end{aligned}
$$

The standard printout for the problem is shown in Table 2.4.6, while information concerning sensitivity analyses is provided in Table 2.4.7.

Table 2.4.6: *Summary of Results* for sample problem

```
                    SUMMARY OF RESULTS

VALUE OF THE OBJECTIVE FUNCTION 10.8000

   DECISION       VALUE AT       OPPORTUNITY
   VARIABLE       OPTIMUM        COST

   X1             4.4000         0.0000
   X2             2.4000         0.0000

   SLACK/EXCESS   CONSTRAINT     OPTIMAL  SHADOW PRICE
   VARIABLE          TYPE        VALUE

   CONSTRAINT 1:     EQ          0.0000      2.2000
   CONSTRAINT 2:     LE          0.0000      0.4000
   CONSTRAINT 3:     GE          9.4000      0.0000
```

The *Summary of Results* shown in Table 2.4.6 subdivides in three parts. The first part shows the optimal value of the objective function , in our example $\bar{z} = 10.8$. The second part shows the optimal values of the decision variables as well as their *opportunity costs*. In our examples, the information provided indicates that at optimum, $\bar{x}_1 = 4.4$ and $\bar{x}_2 = 2.4$. The opportunity cost (sometimes also called *reduced cost*) of a variable indicate how far the price or cost of a variable is away from being included in the optimal solution. In our example, both opportunity costs are zero, as both variables are included in the solution with a positive value. Suppose now that there is a variable in a problem, whose price is \$5 and whose opportunity costs are 3. Furthermore, suppose that the variable equals zero at optimum, i.e., it is not included in the optimal solution with a positive value. The main reason for the variable not being part of the optimal solution is that it is not profitable enough. More specifically, its price is not high enough. The opportunity cost now indicates that the lowest price at which the variable would be included in the solution is its present price plus the opportunity cost, in our example \$5 + \$3 = \$8. In other words, \$8 is the lowest price at which we consider including the variable in the solution. The interpretation of opportunity cost in cost minimization problems is similar. In those problems, it indicates by how much the cost of a variable has to decrease, before it will be part of a solution.

Consider now the third and last part of the *Summary of Results* that is headed the words "slack or excess variable." As discussed earlier in this book, the solver has automatically added a slack variable to the left-hand side of each ≤ inequality and subtracted an excess or surplus variable from the left-hand side of each ≥ inequality.

The printout first provides the number of constraint these variables are added to, followed by a column that specifies the type of constraint (and with it the type of

additional variable) that was automatically added. Here, we use the abbreviations *EQ* for equation, *LE* for a "≤" constraint, and *GE* for a "≥" constraint. The optimal values of the slack, excess, and artificial variables are provided in the next column. In our example, they are 0, 0, and 9.4. These numbers indicate that the variable associated with the first constraint has an optimal value of 0. This is obvious, as the first constraint is an equation, which does not allow left- and right-hand sides to differ. The second constraint is of type "≤," meaning that the system automatically added a slack variable to the left-hand side of the relation. The optimal value of this slack variable is 0, indicating that the constraint is actually satisfied as an equation at optimum, meaning that the constraint is binding at optimum. A constraints that is binding at optimum constitutes a *bottleneck* of the problems. Bottlenecks require special attention as any changes in the right-hand side value of such equations will immediately result in changes of the optimal solution. Finally, the third constraint is of type "≥" and thus had an excess variables subtracted from its left-hand side. The optimal value of this excess variable equals 9.4, indicating that at optimum, the left-hand side exceeded the right-hand side value by 9.4. This is not a bottleneck, so that smaller changes of the third right-hand side value will leave the solution unchanged.

The last column in Table 2.4.6 shows the *shadow prices* associated with the constraints. The shadow price associated with a constraint indicates the change of the value of the objective function, given that the right-hand side value of that constraint increases by one unit. It does not provide any information about how the solution would change, though.

In our example, the shadow price of the first constraint (the equation) equals 2.2. This means that as the first right-hand side value b_1 increases from its original value of 2 by one unit to 3, the value of the objective function increases from 10.8 by 2.2 to 13. Similarly, the increase of the second right-hand side value by one unit from its original value of 16 to 17 will result in a new optimal objective value that is 0.4 higher than the original objective value of 10.8, i.e., $\bar{z} = 11.2$. Finally, the third constraint has a shadow price of 0. This indicates that a change of the third right-hand side value by one unit will neither change the optimal solution, nor the value of the objective function at optimum.

Additional information is available when choosing the *"Sensitivity Analyses"* option. The simulated printout for our example is shown in Table 2.4.7.

The *Sensitivity Analyses* option consists of two parts. The upper part headed by *Coefficients of the Objective Function* analyzes changes of the objective function coefficients c_j, while the part headed by *Right-Hand Side Values* provides information concerning changes of the right-hand side values b_i.

In Table 2.4.7, the line headed by x_1 specifies the original value of $c_1 = 3$, and also the interval = [1, ∞[. This interval indicates that as long as the value of c_1 is equal or larger than 1, the optimal solution shown in Table 2.4.6 will remain optimal.

Table 2.4.7: *Sensitivity Analyses* for sample problem

SENSITIVITY ANALYSES

COEFFICIENTS OF THE OBJECTIVE FUNCTION

VARIABLE	LOWEST ALLOWABLE VALUE	ORIGINAL VALUE	HIGHEST ALLOWABLE VALUE
X1	1.0000	3.0000	INFINITY
X2	−3.0000	−1.0000	INFINITY

RIGHT-HAND SIDE VALUES

CONSTRAINT NUMBER	LOWEST ALLOWABLE VALUE	ORIGINAL VALUE	HIGHEST ALLOWABLE VALUE
CONSTRAINT 1	−1.6154	2.0000	8.0000
CONSTRAINT 2	8.1667	16.0000	INFINITY
CONSTRAINT 3	−INFINITY	15.0000	24.4000

Similarly, we can interpret the numbers in the row headed by x_2. The original value of the objective function coefficient of this variable is $c_2 = -1$. The range specified here is = [–3, ∞[, and it indicates that as long as c_2 is –3 or higher, the solution shown in Table 2.4.6 will remain optimal.

Consider now the bottom part of Table 2.4.7. The row headed by "Constraint 1" specifies the original right-hand side value of the first constraint (which was $b_1 = 2$), as well as the interval [–1.6154, 8.0000]. This interval indicates that as long as the first right-hand side value remains between these bounds, the optimal basis will remain unchanged. In other words, within this range the same constraints remain binding at optimum. The solution, however, may very well change, even within this range. The remaining rows for the other two constraints are interpreted in a similar fashion.

The remainder of this section will present a linear programming problem, formulate it, and then interpret the results shown in a printout. The production planning model we will use throughout this section is as follows.

Example: A footwear manufacturer is planning next year's product line. Part of that line are three types of hiking boots, called "Walker," "Hiker," and "Backpacker." Among the many raw materials used in the production are three particularly important and costly materials: NOwater (a lining that waterproofs the boots), Fabrinsula (fabric insulation for warmth), and Lugster (lug) soles. Currently, 10,000

sq. ft. of NOwater are available at \$20 per sq. ft. Similarly, the manufacturer has access to up to 5,000 oz. of Fabrinsula at \$15 per ounce and up to 7,000 pairs of Lugster soles at \$10 per pair. The requirements for manufacturing the different types of boots are given in the following table.

Table 2.4.8: Data for shoe production problem

	Products			
	Walker	Hiker	Backpacker	
NOwater	1/2	1	$2^1/_2$	[sq. ft. per pair]
Fabrinsula	1/2	2/3	4/3	[oz. per pair]
Lugster	1	1	1	[pair of soles]

Contracts have been signed for the delivery of at least 3,000 pairs of "Walkers," at least 2,000 pairs of "Hikers," and at least 1,000 pairs of "Backpackers." Customers are prepared to purchase additional quantities should they become available. The agreed-upon prices are \$40, \$65, and \$110 per pair of the respective types. (Note: The unit profits in the objective function in the formulation below have been computed as the price minus the costs for the required NOwater, Fabrinsula, and Lugster).

In order to formulate the problem, we first define variables x_j as the quantity of the three types of boots that we made and sell. The formulation of the problem is then as follows.

$$
\begin{array}{lrrrrl}
\text{P: Max } z = & 12.5x_1 & + 25x_2 & + 30x_3 & & \\
\text{s.t.} & \tfrac{1}{2}x_1 + & x_2 + & 2\tfrac{1}{2}x_3 & \leq 10{,}000 & \text{(NOwater availability)} \\
& \tfrac{1}{2}x_1 + & \tfrac{2}{3}x_2 + & 1\tfrac{1}{3}x_3 & \leq 5{,}000 & \text{(Fabrinsula availability)} \\
& x_1 + & x_2 + & x_3 & \leq 7{,}000 & \text{(Lugster soles availability)} \\
& x_1 & & & \geq 3{,}000 & \text{(Walker requirement)} \\
& & x_2 & & \geq 2{,}000 & \text{(Hiker requirement)} \\
& & & x_3 & \geq 1{,}000 & \text{(Backpacker requirement)} \\
& x_1, & x_2, & x_3 & \geq 0. &
\end{array}
$$

Before we provide the printout and continue with our interpretations, some comments regarding the formulation are in order. While the constraints are straightforward, the objective function coefficients have been obtained as follows. Consider the "Walker" boots. They sell for \$40 a pair, from which we have to deduct the costs of ½ sq ft of NOwater (\$10), ½ oz of Fabrinsula (\$7.50) and one pair of soles (\$10), leaving us with a per-unit profit of \$12.50. This is the coefficient found in the above formulation. The unit profits of the other two boots are computed in a similar fashion.

Tables 2.4.9 and 2.4.10 provide the usual printouts with the sensitivity option.

Table 2.4.9: *Summary of Results* for shoe production problem

SUMMARY OF RESULTS

VALUE OF THE OBJECTIVE FUNCTION 143,749.60

DECISION VARIABLE	VALUE AT OPTIMUM	OPPORTUNITY COST
WALKER	3,000.0000	0.0000
HIKER	2,750.0750	0.0000
BACKPACKER	1,249.9249	0.0000

SLACK/EXCESS VARIABLE	CONSTRAINT TYPE	OPTIMAL VALUE	SHADOW PRICE
NOWATER:	LE	2,625.1125	0.0000
FABRINSULA:	LE	0.0000	7.5008
LUGSTER:	LE	0.0000	19.9993
WALKER:	GE	0.0000	-11.2496
HIKER:	GE	750.0750	0.0000
BACKPACKER:	GE	249.9250	0.0000

Table 2.4.10: *Sensitivity Analyses* for shoe production problem

SENSITIVITY ANALYSES

COEFFICIENTS OF THE OBJECTIVE FUNCTION

VARIABLE	LOWEST ALLOWABLE VALUE	ORIGINAL VALUE	HIGHEST ALLOWABLE VALUE
WALKER	−INFINITY	12.50	23.75
HIKER	16.00	25.00	30.00
BACKPACKER	25.00	30.00	50.00

Table 2.4.10 (continued)

RIGHT-HAND SIDE VALUES			
CONSTRAINT	LOWEST ALLOWABLE VALUE	ORIGINAL VALUE	HIGHEST ALLOWABLE VALUE
NOWATER:	7,374.89	10,000.00	INFINITY
FABRINSULA:	4,833.40	5,000.00	5,500.00
LUGSTER:	6,625.00	7,000.00	7,249.89
WALKER:	2,001.60	3,000.00	3,600.02
HIKER:	−INFINITY	2,000.00	2,750.08
BACKPACKER:	−INFINITY	1,000.00	1,249.93

The following is a simulated dialog between an analyst, who modeled the problem and provides the information from the printouts and the decision maker, who asks the questions that are of managerial interest.

Q1: How many pairs of the boots should we manufacture and what will be the associated profit?

A1: The solution suggests that we make 3,000 pairs of Walkers, 2,750 pairs of Hikers, and 1,250 pairs of Backpackers. Given this production plan, we can expect a profit of $ 143,750.

Q2: You are undoubtedly aware of the fact that NOwater, Fabrinsula, and Lugster are critical resources. How many of these do we use in the suggested plan and how much is left over?

A2: The printout tells me that we will have 2,625 sq ft of NOwater left over, while Fabrinsula and Lugster soles are both completely used. The latter two are obvious bottlenecks in the process. In other words, we are using 7,375 sq ft of NOwater, the complete supply of 5,000 oz of Fabrinsula, and all of the 7,000 pairs of Lugster soles.

Q3: I just was informed by our marketing research group that the Hiker boots are very popular and just about all of our customers would be prepared to pay an additional $10 to get a pair of these. Would such a price hike change the optimal solution?

A3 (aside to himself): Ahh, a sensitivity analysis on the objective function coefficient c_2. The range within which the present optimal solution remains optimal is [16, 30]. A price increase by $10 leads to a price of $35 rather than the original $25, which is not in the interval, thus the solution will change.

(to the Decision Maker): If we hike the price by more than $5, our solution will change.

Q4: We are presently making 3,000 Walkers, just enough to satisfy one of our requirements. I wonder if it would be worthwhile to lower the price. Would that lead to increased sales?

A4 (to himself): Another sensitivity analysis on an objective function coefficient. Given the present unit profit of $12.50 for a pair of Walkers, the Sensitivyity Analyses part of the printout indicates that as long as the unit profit remains in the interval $]-\infty, 23.75]$, the optimal solution will not change.

(to the Decision maker): It does not matter by how much you decrease the price of Walkers, we will not sell any more.

Q5: All right then. An interesting thing happened the other day. I met a salesman who offered me an additional 100 oz of Fabrinsula for $8.50 per ounce. Should we purchase that? You told me that we used up all the Fabrinsula that we have. What if I squeeze the man a bit and get it for $5? Will we take it for that price? And what will happen to our profit?

A5 (to himself): A sensitivity analysis on the second right-hand side value b_2. The Summary of Results tells me that the shadow price of Fabrinsula is $7.50, so we should not purchase it for more than that. Also, the basis will not change if we buy up to 500 ounces of Fabrinsula.

(to the Decision Maker): No, don't buy it for $8.50. As a matter of fact, you should not pay more than $7.50 for an extra ounce of Fabrinsula. If you can get it for $5, you can buy up to 500 oz of it. For each ounce that you get on top of what we have, our profit will increase by $2.50. Beyond 500 extra ounces, though, I will have to make additional calculations.

Q6: Wonderful. I have also considered an alternative, though. In one of our trade magazines, I just read an offer for Lugster Soles that we could get for $19 a pair. Would you consider that?

A6 (to himself): The shadow price for Lugster Soles is $20, and the Sensitivity Analyses tell me that this holds for an increase of up to 250 pairs.

(to the Decision Maker): If you can get a pair for $19, get them. They are worth $20 to us, so for each extra pair, we make $1 in addition to our usual profit. You can get up to 250 pairs.

Q7: Thank you so much, Mr. Analyst. Your advice was very helpful. (Putters around with his cell phone). Wait a minute—stop the presses! Our production manager just texted me that 200 pairs of Lugster soles have been damaged in our warehouse and can no longer be used. What are we going to do? How much is that going to cost me?

A7 (to himself): The optimal basis remains unchanged as long as we have between 6,625 and 7,250 pairs of soles. So 200 pairs less will not change the basis, but it will change the solution. And since the shadow price for Lugster Soles is $19, our profit will decrease by 200(19) = $3,600.

The Decision Maker: I knew it. Why are they doing this to me? (Disappears into the bowels of the Administration Building).

Exercises

Problem 1 (a diet problem): A planner considers designing a diet that consists of Coke, garlic fingers, spring rolls, pita pockets, and apples. As far as nutrients go, the planner includes calories, riboflavin, and vitamin A, of which at most 1,500, at least 98, and at least 27 must be included in the diet. The cost minimization problem was subsequently formulated as follows:

$$\text{Min } z = .9x_1 + 2.8x_2 + 3.2x_3 + 5.6x_4 + 3.6x_5$$

$$\begin{aligned} \text{s.t.} \quad & 80x_1 + 310x_2 + 340x_3 + 460x_4 + 20x_5 \le 1{,}500 \\ & 2x_1 + 9x_2 + 25x_3 + 16x_4 + 5x_5 \ge 98 \\ & 5x_2 + 4x_3 + 3x_4 + 10x_5 \ge 27 \\ & x_1, \; x_2, \; x_3, \; x_4, \; x_5 \ge 0. \end{aligned}$$

The printout of the problem is shown in Table 2.4.11.

Table 2.4.11: *Summary of Results* for Problem 1

SUMMARY OF RESULTS

VALUE OF THE OBJECTIVE FUNCTION 16.18609

DECISION VARIABLE	VALUE AT OPTIMUM	OPPORTUNITY COST
COKE	0.0000	0.7470
GARLIC FINGERS	0.0000	0.5026
SPRING ROLLS	3.6739	0.0000
PITA POCKETS	0.0000	3.4104
APPLES	1.2304	0.0000

SLACK/EXCESS VARIABLE	CONSTRAINT TYPE	OPTIMAL VALUE	SHADOW PRICE
CALORIES:	LE	226.2609	0.0000
RIBOFLAVIN:	GE	0.0000	−0.0765
VITAMIN A:	GE	0.0000	−0.3217

(a) What does the diet of the planner consist of? (Indicate type of food and quantity).
(b) How much does the diet cost?
(c) How many calories does the diet include? How much riboflavin? How much vitamin A?
(d) What is the highest price that will put Coke into the solution?
(e) What is the effect of increasing the riboflavin requirement from 98 to 99?

Solution:
(a) The diet consists of no Coke, no garlic fingers, 3.67 spring rolls, no pita pockets, and 1.23 apples.
(b) The cost of the diet is \$16.19.
(c) The diet includes 1,500 – 226.26 = 1,273.74 calories, 98 + 0 = 98 units of riboflavin, and 27 + 0 = 27 units of vitamin A.
(d) Presently, Coke costs 90¢ which is apparently too much for it to be included in the solution. Its opportunity cost is 74.7¢, so that its price will have be decrease by at least that amount. In other words, Coke will be included in the diet if its price is no higher than 90 – 74.7 = 15.3¢.
(e) The cost will increase by 7.65¢.

Problem 2 (an investment problem): An investment agency has been asked to advise one of its clients how to invest all of his \$100,000 among the 4 assets shown in Table 2.4.12.

Table 2.4.12: Risk and rate of return for Problem 2

Assets	Units of risk per dollar invested	Expected rate of return
Northern Mines Shares	4	.15
Bucklin Automobiles	3.5	.11
Royal Bank of Commerce Shares	2	.07
NB Savings Bonds	1	.05

The client would like as high an annual return as is possible to receive while incurring of an average of no more than 2.5 risk units per dollar invested. The amount invested in Royal Bank cannot exceed \$40,000. Furthermore, the investment in Automobiles and Banks combined must be at least \$20,000.

Defining x_1, x_2, x_3, and x_4 for the amount invested in the four alternatives, the problem can be formulated as follows.

P: Max $z = .15x_1 + .11x_2 + .07x_3 + .05x_4$
s.t. $1.5x_1 + x_2 - .5x_3 - 1.5x_4 \leq 0$
$x_1 + x_2 + x_3 + x_4 \leq 100{,}000$
$x_3 \leq 40{,}000$
$x_2 + x_3 \geq 20{,}000$
$x_1, x_2, x_3, x_4 \geq 0$.

The printout is shown in Tables 2.4.13 and 2.4.14.

Table 2.4.13: *Summary of Results* for Problem 2

SUMMARY OF RESULTS

VALUE OF THE OBJECTIVE FUNCTION 9,733.333

DECISION VARIABLE	VALUE AT OPTIMUM	OPPORTUNITY COST
MINES	43,333.33	0.0000
BUCKLIN	0.0000	0.01
ROYAL BANK	20,000.00	0.0000
SAVINGS	36,666.67	0.0000

SLACK/EXCESS VARIABLE	CONSTRAINT TYPE	OPTIMAL VALUE	SHADOW PRICE
RISK	LE	0.0000	0.0333
INVESTMENT	LE	0.0000	0.1000
BANK LIMIT	LE	20,000.00	0.0000
BANK & AUTO	GE	0.0000	−0.0133

Table 2.4.14: *Sensitivity Analyses* for Problem 2

SENSITIVITY ANALYSES

COEFFICIENTS OF THE OBJECTIVE FUNCTION

VARIABLE	LOWEST ALLOWABLE VALUE	ORIGINAL VALUE	HIGHEST ALLOWABLE VALUE
MINES	0.1300	0.1500	INFINITY
BUCKLIN	−INFINITY	0.1100	0.1200
ROYAL BANK	0.0600	0.0700	0.0833
SAVINGS	0.0300	0.0500	0.0700

Table 2.4.14 (continued)

RIGHT-HAND SIDE VALUES			
CONSTRAINT	LOWEST ALLOWABLE VALUE	ORIGINAL VALUE	HIGHEST ALLOWABLE VALUE
RISK	−130,000.00	0.0000	110,000.00
INVESTMENT	26,666.67	100,000.00	INFINITY
BANK LIMIT	20,000.00	40,000.00	INFINITY
BANK & AUTO	0.00	20,000.00	40,000.00

(a) How much is invested in each of the alternatives and what is the average rate of return?
(b) Identify the bottlenecks in the investment plan.
(c) If we could borrow some additional funds at 11%, would it be worth our while? Explain in one short sentence.
(d) There is a rumor that the return of the Royal Bank will increase to 8%. Will this change the investor's plans? What if it decreases to 5.5%? Explain in one short sentence.

Solution: (a) The solution prescribes investments as follows: Mines: \$43,333.33, Automobile: \$0, Bank: \$20,000, and NB Savings: \$36,666.66. The average rate of return is 9.7333%.
(b) Risk constraint (constraint 1), total investment level (constraint 2), and at least \$20,000 in "Auto and Bank" (constraint 4) are all tight.
(c) Shadow price of total investment (constraint 2) is 0.1, i.e., the benefit of an extra dollar is 10¢. Given the interest rate of 11% on loans, it is not worthwhile to borrow.
(d) Sensitivity on objective function coefficients. The range for the bank shares extends from 6% to 8.33%, so an increase from 7% to 8% will not change the investment plan. A decrease to 5.5% will.

Problem 3 (a transportation problem): Consider a transportation problem with two origins (warehouses) and three destinations (customers). The supplies at the warehouses are 60 and 80 units, respectively, while the demand is exactly 30, 50, and 40, respectively. The problem has been formulated as follows:

$$\text{Min } z = 3x_{11} + 7x_{12} + 4x_{13} + 9x_{21} + 2x_{22} + 5x_{23}$$
$$\text{s.t. } x_{11} + x_{12} + x_{13} \le 60$$
$$x_{21} + x_{22} + x_{23} \le 80$$
$$x_{11} + x_{21} = 30$$
$$x_{12} + x_{22} = 50$$
$$x_{13} + x_{23} = 40$$
$$x_{11}, x_{12}, x_{13}, x_{21}, x_{22}, x_{23} \ge 0.$$

The printout is shown in Tables 2.4.15 and 2.4.16.

Table 2.4.15: *Summary of Results* for Problem 3

SUMMARY OF RESULTS

VALUE OF THE OBJECTIVE FUNCTION 360.0000

DECISION VARIABLE	VALUE AT OPTIMUM	OPPORTUNITY COST
X11	30.0000	0.0000
X12	0.0000	6.0000
X13	30.0000	0.0000
X21	0.0000	5.0000
X22	50.0000	0.0000
X23	10.0000	0.0000

SLACK/EXCESS VARIABLE	CONSTRAINT TYPE	OPTIMAL VALUE	SHADOW PRICE
ORIGIN 1	LE	0.0000	1.0000
ORIGIN 2	LE	20.0000	0.0000
DESTINATION 1	EQ	0.0000	-4.0000
DESTINATION 2	EQ	0.0000	-2.0000
DESTINATION 3	EQ	0.0000	-5.0000

Table 2.4.16: *Sensitivity Analyses* for the Problem 3

SENSITIVITY ANALYSES

COEFFICIENTS OF THE OBJECTIVE FUNCTION

VARIABLE	LOWEST ALLOWABLE VALUE	ORIGINAL VALUE	HIGHEST ALLOWABLE VALUE
11	−INFINITY	3.00	8.00
X12	1.00	7.00	INFINITY
X13	−1.00	4.00	5.00
X21	4.00	9.00	INFINITY
X22	-INFINITY	2.00	8.00
X23	4.00	5.00	10.00

Table 2.4.16 (continued)

RIGHT-HAND SIDE VALUES			
CONSTRAINT	LOWEST ALLOWABLE VALUE	ORIGINAL VALUE	HIGHEST ALLOWABLE VALUE
ORIGIN 1	40.00	60.00	70
ORIGIN 2	60.00	80.00	INFINITY
DESTINATION 1	20.00	30.00	50.00
DESTINATION 2	0.00	50.00	70.00
DESTINATION 3	30.00	40.00	60.00

(a) What is the shipment plan and what are the associated costs?
(b) Which of the warehouses are fully used and which have still some units of the product in them (and how many)?
(c) What if the per-unit-cost of a shipment from origin 2 to destination 3 were to increase by \$2, would that change the optimal solution? What if the cost were to decrease by \$2?
(d) What would happen if the number of units available at the first warehouse were to be reduced by one unit? (Include cost considerations).
(e) What if we were offered extra units delivered to warehouse 2 at a rate of \$2 per unit?

Solution: (a) The shipments from the first origin 1 to the three destinations are 30, 0, and 30, while the shipments from Origin 2 to the three destinations are 0, 50, and 10. The total transportation costs are \$360.
(b) All units in warehouse 1 are shipped out, but 20 units are left over in warehouse 2.
(c) The range for the unit transportation costs c_{23} is [4, 10]. An increase by \$2 puts the unit transportation cost at 5 + 2 = \$7, which is in the interval. As a result, there will be no change in the optimal transportation plan, but the costs will increase by $2x_{23} = 20$. A decrease of \$2 would put the unit transportation cost at 5 – 2 = 3, which is outside of the interval, so that the optimal transportation plan (and its costs) will change.
(d) As the constraint the belongs to Origin 1 has zero slack, the solution will change. As the shadow price is 1, the total cost will increase by \$1.
(e) There are still 20 units left at Origin 2, so that no additional units are needed, and we decline the offer.

2.5 Duality

This section explores some aspects of duality theory, the theory behind linear programming that explores how and why solution methods work. Given the scope of this book, we will restrict ourselves to some of the relations and interpretations,

without getting into technical details. For more details, interested readers are referred to the standard advanced texts such as Dantzig (1963), Dantzig and Thapa (1997), or Eiselt and Sandblom (2007).

In order to simplify our discussion, we will base our arguments on a production problem similar to that presented in Section 2.2.1 when we introduced linear programming. In this case, there are two products, floor boards and spindles, that are processed on three machines: a saw, a router, and a sander. The unit profits of the two products are \$1 per floor board and \$4 per spindle. The machines have capacities (in seconds) of 50,000, 450,000, and 600,000. It takes 5 seconds to saw the floor board or the spindle. Thirty seconds are needed to process a floor board on the router, while a spindle takes 90 seconds on this machine. One floor board requires 20 seconds on the sander, while a spindle needs 100 seconds for processing.

Defining variables x_1 and x_2 as the number of floor boards and spindles that we make and sell, respectively, we can formulate the problem as follows.

$$
\begin{array}{lrcll}
\text{P: Max } z = & 1x_1 + & 4x_2 & & \text{(max profit)} \\
\text{s.t.} & 5x_1 + & 5x_2 \leq & 50{,}000 & \text{(saw)} \\
& 30x_1 + & 90x_2 \leq & 450{,}000 & \text{(router)} \\
& 20x_1 + & 100x_2 \leq & 600{,}000 & \text{(sander)} \\
& x_1, & x_2 \geq & 0. & \text{(nonnegativity)}
\end{array}
$$

To every linear programming problem, which we will call a *primal problem*, we can now assign another linear programming problem, called the *dual problem*. The variables in the primal problem are also referred to as the *primal variables*.

In order to explain the dual problem, we will refer to the firm that has the primal problem as its planning model as the *Manufacturer*. Suppose now that there is another company that, for reasons to be come clear as we proceed, we will refer to as the *Lessor*. The *Lessor* actually owns the saws, routers, and sanders, which he can use to either manufacture floor boards and spindles himself or lease out the machine time to the *Manufacturer* or Lessee. The task for the *Manufacturer* is now to set up a pricing system that will minimize its own overall costs, while making it interesting to the *Lessor* to rent machine time to the *Manufacturer* rather than make the products himself. Note that the *Lessor* will not rent out an hour here or there, he either rents the entire time—50,000 seconds on the saw, 450,000 seconds on the router, and 600,000 seconds on the sander—or not at all.

In order to determine such a pricing system, the *Manufacturer* will set up a pricing system that defines u_1 as the price per second on the saw, u_2 as the price per second on the router, and u_3 as the price per second on the sander. Clearly, the *Manufacturer's* task is to minimize the total cost of leasing the equipment. Leasing the machine time on the saw will cost u_1 dollars per second, and the time to lease is 50,000 seconds, and similar for the other two machines. Hence the objective is to minimize $50{,}000u_1 + 450{,}000u_2 + 600{,}000u_3$. The next task is to

make it interesting to the *Lessor* to rent out the machine time rather than manufacturing it himself. This can be achieved by ensuring that the price achieved for the rental time that is required to make one unit of a product is at least as much as making and selling the product directly. In order to illustrate, consider the floor boards. It takes 5 seconds on the saw to make it, which, given the pricing system, is evaluated at $5u_1$. In addition, we also use time on the router and the sander, and these times are evaluated at $30u_2$ and $20u_3$, respectively. The sum of all of these times—the rental time equivalent to making one floor board—should be at least as much as the profit of making a floor board. If it were not, the *Lessor* might as well go into the manufacturing business himself. This means that we have to require that $5u_1 + 30u_2 + 20u_3 \geq 1$. A similar constraint needs to be written for the spindles, again in order to make it as least as attractive for the *Lessor* to rent out time rather than manufacture himself. The constraint is $5u_1 + 90u_2 + 100u_3 \geq 4$. Adding the fact that the prices must all be nonnegative, we now have formulated a complete dual problem, which we will call P_D. Below, we show the primal problem and the dual problem next to each other.

P: Max $z = 1x_1 + 4x_2$	P_D: Min $z_D = 50{,}000u_1 + 450{,}000u_2 + 600{,}000u_3$
s.t. $5x_1 + 5x_2 \leq 50{,}000$	s.t. $5u_1 + 30u_2 + 20u_3 \geq 1$
$30x_1 + 90x_2 \leq 450{,}000$	$5u_1 + 90u_2 + 100u_3 \geq 4$
$20x_1 + 100x_2 \leq 600{,}000$	$u_1, u_2, u_3 \geq 0$.
$x_1, x_2 \geq 0$.	

Before we continue discussing the relations between a primal and its associated dual problem, we would like to point out that the roles of variables and constraints in the two problems are exchanged. The objective function coefficients of the primal are found on the right-hand sides of the dual, while the right-hand side values of the primal are in the objective function of the dual. Similarly, the technological coefficients on the left-hand sides are the same, but with rows and columns exchanged.

Tables 2.5.1a and 2.5.1b show the optimal solutions of the primal and dual problem, respectively. While it is optimal for the *Manufacturer* to make no floor boards and 5,000 spindles (for a total profit of \$20,000), the Lessor should charge nothing for the saw and the sander, but 4.44¢ for each second of the router. Given that he leases the machines to the *Manufacturer* rather than making floor boards and spindles himself, his profit from the leasing will be \$20,000.

Comparing the primal and dual solutions, we also note that the optimal values of the primal variables are found in the dual solution as shadow prices of resources, while the optimal values of the primal slack and excess variables are the opportunity costs of the dual variables. Similarly, the shadow prices of the primal resources equal the optimal values of the dual decision variables, and the opportunity costs of the primal variables equal the optimal values of the slack and excess variables in the dual. This means, of course, that we can solve either the primal or the dual solution and have both optimal solutions available in one Summary of Results.

In order to explain some further relations between the primal and its associated dual problem, multiply each primal constraint with its associated dual variable and each dual constraint with the primal variable it is associated with. (Note that due to the nonnegativity constraints on the variables, the inequalities do not change). We then obtain the following system:

$z = 1x_1 + 4x_2$ (the primal objective function)

$5x_1u_1 + 5x_2u_1 \leq 50{,}000u_1$ (the first primal constraint multiplied by u_1)
$30x_1u_2 + 90x_2u_2 \leq 450{,}000u_2$ (the second primal constraint multiplied by u_2)
$20x_1u_3 + 100x_2u_3 \leq 600{,}000u_3$ (the third primal constraint multiplied by u_3)
$5u_1x_1 + 30u_2x_1 + 20u_3x_1 \geq 1x_1$ (the first dual constraint multiplied by x_1)
$5u_1x_2 + 90u_2x_2 + 100u_3x_2 \geq 4x_2$ (the second dual constraint multiplied by x_1)
$z_D = 50{,}000u_1 + 450{,}000u_2 + 600{,}000u_3$ (the dual objective function)

$x_1, x_2 \geq 0, u_1, u_2, u_3 \geq 0.$ (nonnegativity constraints)

Adding all primal constraints we obtain $5x_1u_1 + 5x_2u_1 + 30x_1u_2 + 90x_2u_2 + 20x_1u_3 + 100x_2u_3 \leq 50{,}000u_1 + 450{,}000u_2 + 600{,}000u_3$, where we note that the right-hand side value equals z_D. Similarly, adding all dual constraints, we obtain $5u_1x_1 + 30u_2x_1 + 20u_3x_1 + 5u_1x_2 + 90u_2x_2 + 100u_3x_2 \geq 1x_1 + 4x_2$, where we note that the right-hand side value of the aggregate constraint equals the primal value of the objective function z. Putting the two expressions together, we obtain $z_D = 50{,}000u_1 + 450{,}000u_2 + 600{,}000u_3 \geq 1x_1 + 4x_2 = z$, or simply $z_D \geq z$. This property is commonly referred to as *weak duality*. Given that the primal problem maximizes z and the dual problem minimizes z_D, the two values will move towards each other and be equal when they reach an optimal solution. This situation is shown in Figure 2.5.1.

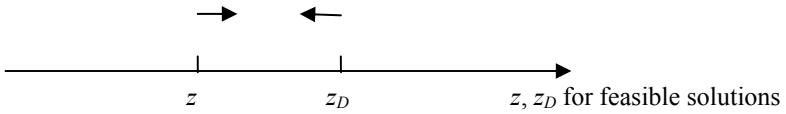


Figure 2.5.1

Table 2.5.1: Optimal solution of primal and dual problem

SUMMARY OF RESULTS - PRIMAL

VALUE OF THE OBJECTIVE FUNCTION 20,000.00

DECISION VARIABLE	VALUE AT OPTIMUM	OPPORTUNITY COST
FLOOR BOARDS	0.0000	0.3333
SPINDLES	5,000.00	0.0000

SLACK/EXCESS VARIABLE	OPTIMAL VALUE	SHADOW PRICE
SAW	25,000.00	0.0000
ROUTER	0.0000	0.0444
SANDER	100,000.00	0.0000

(a)

SUMMARY OF RESULTS - DUAL

VALUE OF THE OBJECTIVE FUNCTION 20,000.00

DECISION VARIABLE	VALUE AT OPTIMUM	OPPORTUNITY COST
PRICE, SAW	0.0000	25,000.00
PRICE, ROUTER	0.0444	0.0000
PRICE, SANDER	0.0000	100,000.00

SLACK/EXCESS VARIABLE	OPTIMAL VALUE	SHADOW PRICE
FLOOR BOARDS	0.3333	0.0000
SPINDLES	0.0000	5,000.00

(b)

In order for the two objective values to be equal at optimum, it is necessary that *all* of the primal inequalities multiplied by their respective dual variables and all dual constraints multiplied by their respective primal variables are satisfied as equations. In our example, this means that if $\bar{x}_1, \bar{x}_2$, $\bar{x}_3$ is an optimal solution of the primal problem and $\bar{u}_1$, $\bar{u}_2$ is an optimal solution of the dual problem, then

$5\bar{x}_1\bar{u}_1 + 30\bar{x}_2\bar{u}_1 = 50{,}000\bar{u}_1$,
$30\bar{x}_1\bar{u}_2 + 90\bar{x}_2\bar{u}_2 = 450{,}000u_2$, and
$20\bar{x}_1\bar{u}_3 + 100\bar{x}_2\bar{u}_3 = 600{,}000\bar{u}_3$ for the primal constraints, and

$50\bar{u}_1\bar{x}_1 + 30\bar{u}_2\bar{x}_1 + 20\bar{u}_3\bar{x}_1 = 1\bar{x}_1$, and
$5\bar{u}_1\bar{x}_2 + 90\bar{u}_2\bar{x}_2 + 100\bar{u}_3\bar{x}_2 = 4\bar{x}_2$ for the dual constraints.

If we were to define slack variables S_1, S_2, and S_3 for the three primal constraints and excess variables E_1^D and E_2^D for the two dual constraints, and their optimal values are also indicated by a bar over the variables, we can rewrite the above equations as $\bar{S}_1\bar{u}_1 = 0$, $\bar{S}_2\bar{u}_2 = 0$, and $\bar{S}_3\bar{u}_3 = 0$ for the primal problem, and $\bar{E}_1^D\bar{x}_1 = 0$ and $\bar{E}_2^D\bar{x}_2 = 0$ for the dual problem. These conditions are usually called (*weak*) *complementary slackness conditions*.

These conditions mean that if an inequality constraint is not satisfied as an equation at optimum, then its dual variable must be equal to zero. If a constraint is satisfied as an equation, then its dual variable may be zero or positive. In terms of our example, this means that if we do not fully use a resource (here: machine capacities), then the dual variable (the shadow price of the resource) must equal zero. If, on the other hand, a resource is fully used, then the dual variable may be positive. Similarly, if a variable is positive, then its opportunity cost must be zero; if a variable is zero, then its opportunity cost may be positive. This corresponds with the interpretation we provided in Section 2.4 on postoptimality analyses.

It is worth noting that the dual of the dual problem is again the primal problem.

The remainder of this section will demonstrate some further relations between a pair of primal and dual problems. In general, we have three possible cases:

(1) The primal problem and its dual both have finite optimal solutions with $\bar{z} = \bar{z}_D$.
(2) One of the two problems has no feasible solution, while the other has unbounded "optimal" solutions.
(3) Both problems have no feasible solutions.

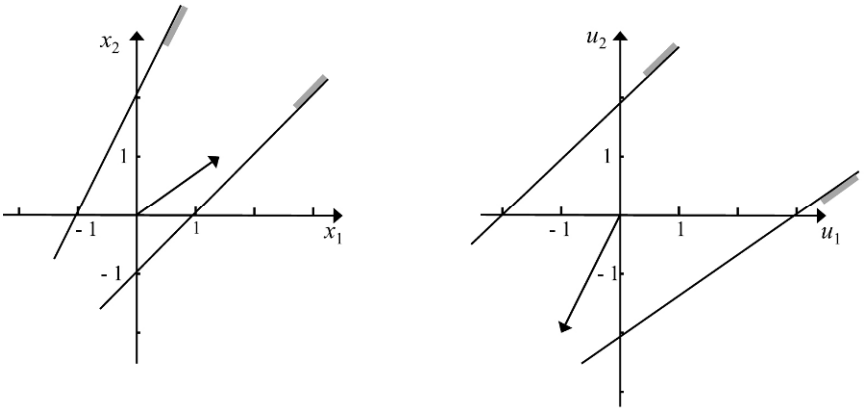
We already have an example of the first case: the numerical illustration used throughout this section belongs into that category.

In order to demonstrate the second case, consider the following pair of dual problems:

P: Max $z = 3x_1 + 2x_2$
s.t. $x_1 - x_2 \le 1$
$-2x_1 + 1x_2 \le 2$
$x_1, \quad x_2 \ge 0$

P_D: Min $z_D = -u_1 + 2u_2$
s.t. $u_1 - 2u_2 \ge 3$
$-u_1 + u_2 \ge 2$
$u_1, \quad u_2 \ge 0.$

The graphical representations are shown in Figure 2.5.2a and b. It is apparent that while the primal problem has unbounded "optimal" solutions, its dual problem has no feasible solution.



(a) (b)

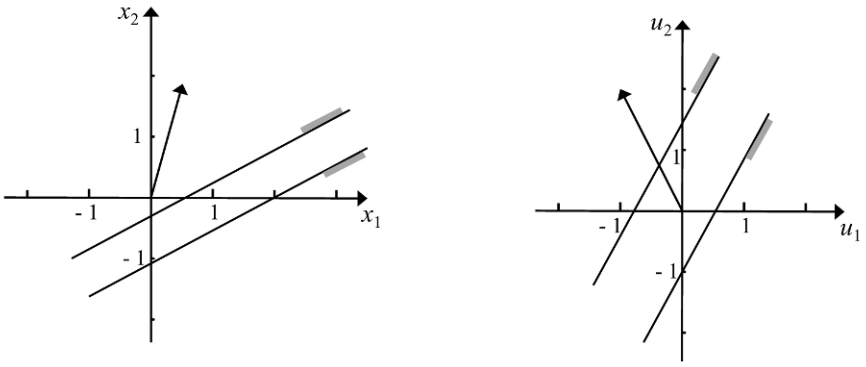
Figure 2.5.2

Finally the third case, in which neither problem has a feasible solution. As a numerical example, consider the following pair of dual problems:

P: Max $z = x_1 + 3x_2$
s.t. $2x_1 - 4x_2 \le 1$
$-x_1 + 2x_2 \le -2$
$x_1, \quad x_2 \ge 0$

P_D: Min $z_D = u_1 - 2u_2$
s.t. $2u_1 - u_2 \ge 1$
$-4u_1 + 2u_2 \ge 3$
$u_1, \quad u_2 \ge 0.$

The graphical representation of the two problems is shown in Figure 2.5.3. It is apparent that the constraints in the primal and the dual problems are parallel to each other, so that the feasible set is empty.



(a) (b)

Figure 2.5.3

Exercises

Problem 1 (setting up the dual problem): Consider the following primal linear programming problem P:

$$\begin{aligned} \text{P: Min } z &= -3x_1 + 4x_2 \\ \text{s.t.} \quad x_1 &+ 2x_2 = 5 \\ 5x_1 &- x_2 \geq 2 \\ x_1, &\quad x_2 \geq 0. \end{aligned}$$

Set up the dual problem P_D.

Solution: Since we have not provided any rules for the transformation other than for maximization problems with "≤" constraints, so we simply bring the problem P into that form. Changing a minimization problem to a maximization problem and changing the direction of an inequality is standard. As far as the equation $x_1 + 2x_2 = 5$ is concerned, we replace it by two inequalities $x_1 + 2x_2 \leq 5$ *and* $x_1 + 2x_2 \geq 5$. We can then define an equivalent version of the problem description P as

$$\begin{aligned} \text{P': Max } -z &= 3x_1 - 4x_2 \\ \text{s.t.} \quad x_1 &+ 2x_2 \leq \ 5 \\ -x_1 &- 2x_2 \leq -5 \\ -5x_1 &+ \ x_2 \leq -2 \\ x_1, &\quad x_2 \geq 0. \end{aligned}$$

The dual of this problem is then

$$
\begin{aligned}
\mathrm{P}'_{\mathrm{D}}: \ & \text{Min } -z_D = 5u_1 - 5u_2 - 2u_3 \\
\text{s.t.} \quad & u_1 - u_2 - 5u_3 \geq 3 \\
& 2u_1 - 2u_2 + u_3 \geq -4 \\
& u_1, \quad u_2 \quad \geq 0.
\end{aligned}
$$

If desired, we can clean up this problem a bit and write it in its equivalent form

$$
\begin{aligned}
\mathrm{P_D}: & \text{ Max } z_D = -5u_1 + 5u_2 + 2u_3 \\
\text{s.t.} \quad & u_1 - u_2 - 5u_3 \geq 3 \\
& -2u_1 + 2u_2 - u_3 \leq 4 \\
& u_1, \quad u_2 \quad \geq 0.
\end{aligned}
$$

Problem 2 (duality for a single-constraint primal problem): Consider the following single-constraint primal linear programming problem:

$$
\begin{aligned}
\mathrm{P}: & \text{ Max } z = 2x_1 + 3x_2 + 8x_3 + 7x_4 + 6x_5 \\
\text{s.t.} \quad & x_1 + 6x_2 + 4x_3 + 3x_4 + 3x_5 \leq 11 \\
& x_1, \quad x_2, \quad x_3, \quad x_4, \quad x_5 \geq 0.
\end{aligned}
$$

(a) Formulate the dual problem $\mathrm{P_D}$. How many variables does it have?
(b) Show that $\mathrm{P_D}$ can be solved by simple inspection. State the optimal solution and objective function value of $\mathrm{P_D}$. Explain why $\bar{z} = \bar{z}_D$.
(c) Using duality relationships, find the unique optimal solution to P.

Solution: (a) The problem under consideration is a continuous knapsack problem similar to those discussed in Chapter 4 of this book. Since there is only a single constraint in the primal problem, the dual problem features only a single variable. The dual is

$$
\begin{aligned}
\mathrm{P_D}: & \text{ Min } z_D = 11u \\
\text{s.t.} \quad & u \geq 2 \\
& 6u \geq 3 \\
& 4u \geq 8 \\
& 3u \geq 7 \\
& 3u \geq 6 \\
& u \geq 0.
\end{aligned}
$$

(b) With $z_D = 11u$, the optimal value of the variable u must be as small as possible. Considering the lower bounds specified in the constraints, we determine that $\bar{u} = \max\{2, \frac{3}{6}, \frac{8}{4}, \frac{7}{3}, \frac{6}{3}, 0\} = 2\frac{1}{3}$, so that $\bar{z}_D = 11\bar{u} = 25.6667$. Since P has feasible solutions (for instance $x_1 = x_2 = \ldots = x_5 = 0$), and $\bar{z}_D$ exists, $\bar{z}$ must exist as well and $\bar{z} = \bar{z}_D = 25\frac{2}{3}$.

(c) Since $3u \geq 7$ is the only constraint in the dual problem P_D that is tight (binding) at optimum, the excess variables of all other dual constraints are strictly positive at optimum. The complementary slackness conditions then require that $\bar{x}_1 = \bar{x}_2 = \bar{x}_3 = \bar{x}_5 = 0$. Since $\bar{u} = 2\frac{1}{3} > 0$, the slack variable of the corresponding primal constraint must be zero at optimum, again due to complementary slackness. The primal constraint is then binding at optimum, so that $3\bar{x}_4 = 11$ or $\bar{x}_4 = 3.6667$.

3 Multiobjective Programming

As diverse as the problems in the previous chapters have been, they share one common feature: they all have one single objective function and the result is an optimal solution (or multiple optima, in case of dual degeneracy). However, the concept of optimality applies only in case of a single objective. If we state that something is "the best" or optimal, we always have an objective in mind: the fastest car, the most comfortable vehicle, the automobile that is cheapest to operate, and so forth. Whenever a second or even more objectives are included in a problem, the concept of optimality no longer applies. For instance, if the top speed of a vehicle and its gas mileage are relevant concerns, then the comparison between a car, whose speed may go up to 110 miles per hour and which gives 20 miles to the gallon (highway rating) and a vehicle that can go up to 90 miles per hour and which gives 25 miles to the gallon is no longer a simple one: the former car is faster at the expense of fuel efficiency. It will now depend on the decision maker which of the two criteria is considered more important. In other words, the decision maker will—sooner or later—have to specify a *tradeoff* between the criteria. This is the type of problems considered in this chapter.

While the terminology in this field is not quite standardized, we typically refer to problems with multiple objectives or evaluation criteria as *multicriteria decision making problems* or *MCDM*. There are two major subclasses of these problems, one called *multi-attribute decision making* (*MADM*), and the other being *multiobjective (linear) programming* (*MOLP*). In simple words, *MADM* problems have a finite number of possible decisions among which the decision maker has to choose one, given a number of criteria. Multiobjective programming problems, on the other hand, are optimization problems like those discussed in the previous chapters, except that they have at least two objective functions. In this chapter we will exclusively deal with *MOLP*.

Interestingly enough, *MOLP* have been around almost as long as linear programming problems. Similarly, the replacement of the concept of optimality by the (much weaker) concept of pareto-optimality has been known to economists for more than hundred years. (It is named after the Italian economist Vilfredo Pareto, 1848-1923). In the sections below, we discuss two of the main approaches to multiobjective

H.A. Eiselt and C.-L. Sandblom, *Operations Research: A Model-Based Approach*,
DOI 10.1007/978-3-642-10326-1_3, © Springer-Verlag Berlin Heidelberg 2010

linear programming: the *vector optimization* problem, and *goal programming* problems. Their main distinction is rooted in the decision maker's input. In vector optimization problems, the decision maker does not provide any input regarding the tradeoff between objectives. This means, of course, that many pareto-optimal solutions will be generated, which then must be compared manually by the decision maker—a process, in which he will have to use some (probably) implicit tradeoffs. Another possibility is to openly define tradeoffs, which then allows the analyst to reduce the problem to a standard linear programming problem. The problem with this approach is that it will be impossible for any decision maker to specify with any degree of certainty that one objective is, say, 2.7 times as important as another. Such an approach will have to rely very heavily on sensitivity analyses. Finally, we present *goal programming*, an approach that blurs the distinction between objectives and constraints.

3.1 Vector Optimization

The most logical way to introduce vector optimization problems starts with linear programming and then extends the analysis to multiple objectives. As usual, we will give preference to intuitive reasoning based on graphical arguments.

As far as the formal statement of a vector optimization problem is concerned, consider the following example that we will use in this in the next section.

$$\begin{aligned}
&\text{Max } z_1 = \ 3x_1 + x_2 \\
&\text{Max } z_2 = -2x_1 + x_2
\end{aligned}$$

$$\begin{aligned}
\text{s.t.} \quad & -x_1 + x_2 \le 3 && (I)\\
& \quad\ x_2 \le 4 && (II)\\
& \ x_1 + x_2 \le 6 && (III)\\
& \ x_1 \quad\ \le 5 && (IV)\\
& \ x_1, \quad x_2 \ge 0.
\end{aligned}$$

First of all, the name "vector optimization" stems from the fact that rather than a single objective, we have a "vector" of objectives.

For now, consider a single objective , whose gradient of the objective function and iso-profit line through some point "x" are shown in Figure 3.1.

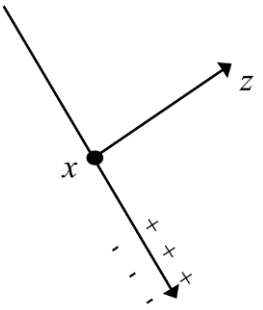


Figure 3.1

To recap from our discussion in linear programming, the "*z*" in Figure 3.1 indicates the gradient of the objective function, while the line with flags "+++" and "---" is the iso-profit line through an arbitrary point *X*. This iso-profit line subdivides the space into two halfspaces: all points in the halfspace flagged with "+++" have objective function values better (i.e., higher for maximization problems and lower for minimization problems) than the point *X*.

Consider now a problem with two objective functions. We can then take an arbitrary point *X*, anchor both objective functions at that point, and plot iso-profit lines through it. This is shown in Figure 3.2.

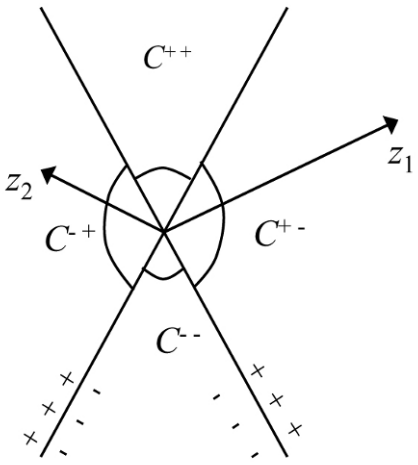


Figure 3.2

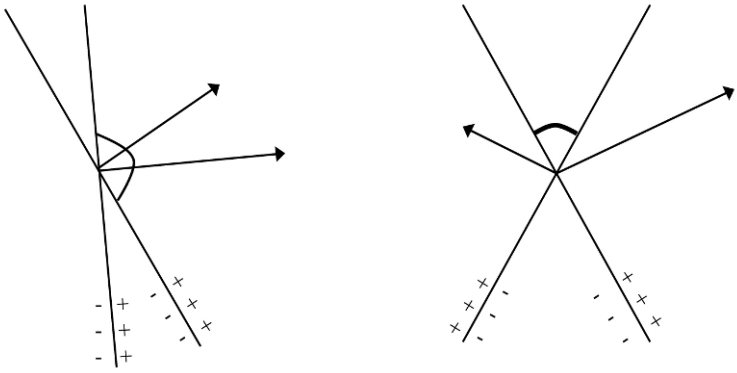
Figure 3.2 is based on a problem with two objectives, whose gradients are shown by the arrows marked with z_1 and z_1, respectively, and their two iso-profit lines.

The two iso-profit lines subdivide the plane into four parts, labeled with C^{++}, C^{+-}, C^{-+}, and C^{--}. In the following analysis, we will compare points in these four parts of the plane with x.

First consider any point in the set labeled C^{--}. As compared to x, this point will be worse than x with respective to the first objective and also worse with respect to the second objective, as it is located in the intersection of the two halfplanes flagged with "– – –. " This means that if we had realized point x, there would be no reason to move to any point in the set C^{--}, as the new point will be worse than x with respect to both objectives. Next, consider a point in the set C^{-+}. Here, things are a bit more difficult, as any point in that set will be worse than x with respect to the first objective but better than x with respect to the second objective. This means that points in C^{-+} are not comparable to x.

A similar argument applies to all points in the set C^{+-}. All points in this set are better than X with respect to the first objective but worse than X with respect to the second objective. So again, no comparison is possible. Finally, consider the set C^{++}. Any point in this set is better than X with respect to both objectives, so that we would move out of the present solution X into C^{++} whenever possible. As all points in C^{++} are better than x, we will call C^{++} the *improvement cone* (rooted at x).

Before using this concept to solve vector optimization problems, it is useful to discuss the relation between objectives. Suppose that two objectives are very similar, for example Max $z_1 = 2x_1 + 5x_2$ and Max $z_2 = 2x_1 + 6x_2$. It is apparent that there is only little conflict between the two. This is shown in Figure 3.3a, where the angle between the two gradients is very small. This results in an improvement cone with a very large angle. Actually, in the extreme case of two identical objectives (i.e., no conflict), the improvement cone is then the same as that shown in Figure 3.1 with the halfplane labeled "+++."



(a) (b)

Figure 3.3

On the other hand, consider two objectives that show extensive conflict. This situation is depicted in Figure 3.3b. Here, the angle between the two gradients is large, and the angle of the improvement cone is very small, indicating that there is only limited potential for improvement. In the limiting case, one gradient would be diametrically opposed to the other, a case of total conflict. In such a case, the improvement cone is empty and there is no potential for any improvement.

However, it may be possible in such a case to find a compromise after all. The main idea is to introduce additional criteria. As an example, consider a couple who is planning this year's vacation. Suppose that the husband would like to maximize the amount of time the couple spends on the beach to relax (and watch other people), the wife would like to minimize the time on the beach and go boating instead, something the husband has absolutely no interest in. (A somewhat similar situation has been dealt with in game theory under the name "battle of the sexes"). This has escalated to a major fight, and as it is, there is no room for compromise. Suppose now that we introduce another activity, e.g., events, festivals, zoos, or museums, which we will call sites and events, something both are interested in, at least to some extent. The objective functions given two dimensions are shown in Figure 3.4, indicating that there is now an actual possibility for compromise. Similar situations occur in labor negotiations, in which management wants to minimize the amount spent on wages and salaries, while labor wants to maximize it. Issues added to the list of topics to be negotiated could include concerns such as work conditions, something both parties are interested in.

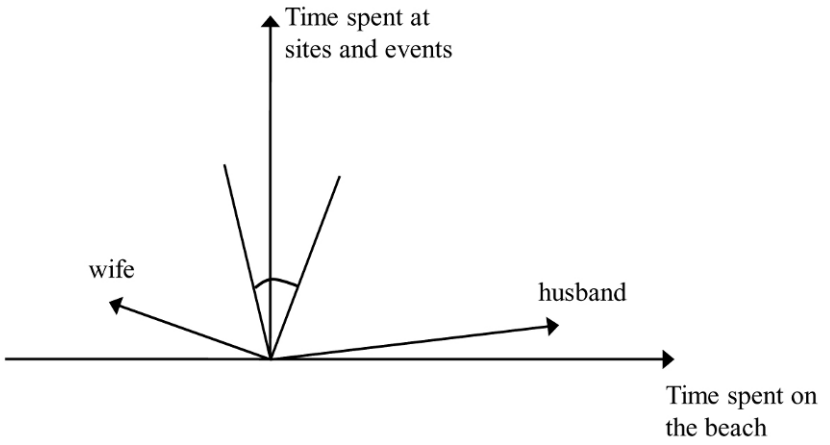


Figure 3.4

Back to the improvement cone. We stated above that whenever possible, we would try to move out of an existing point X into the improvement cone. The only thing that could prevent us from doing so are the constraints. In case it is possible to move out of a feasible point X to another feasible point in the improvement cone rooted at X, we should do so, indicating that the point X is not a point to be

considered further, as there are other points that are better than X with respect to all objectives. In other words, if the intersection of the improvement cone and the feasible set includes at least one more point other than X, then X does no longer have to be considered.

On the other hand, if moving out of X into its improvement cone will always result in the loss of feasibility, then the point X is called *nondominated* (or, alternatively, *noninferior*, *efficient*, or *pareto-optimal*). We will use these terms interchangeably. The collection of all nondominated points is called the efficient frontier. It is apparent that points in the interior of the feasible set cannot be nondominated, as it is possible to move out of them into any direction and stay feasible, at least for some distance. This is shown in Figure 3.5, in which the feasible set is shaded, and the extreme points are *0*, *A*, *B*, *C*, *D*, and *E*. The gradients of the two objectives are labeled z_1 and z_2, and improvement cones are shown anchored at all extreme points as well as at some interior point *X*. From any of these points, we will try to move into its improvement code and determine whether or not it is possible to do so and stay within the feasible region. Clearly, at the interior point *X*, this is feasible.

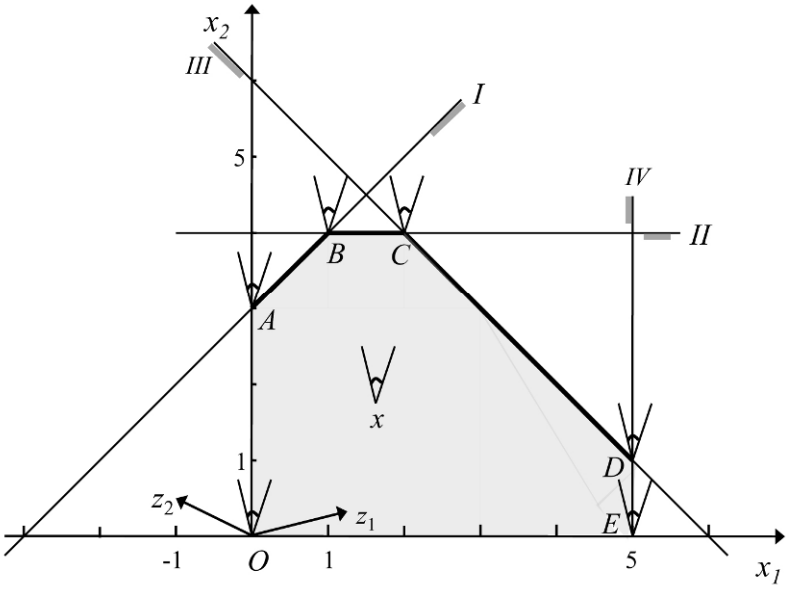


Figure 3.5

Consider now the extreme points of the feasible set. Starting with the origin *0*, we see that it is possible to stay feasible by keeping to the right of the improvement cone, consequently the point 0 is dominated. The situation is different for the extreme points *A*, *B*, *C*, and *D*. All of these points have improvement cones whose intersections with the feasible set equal just the point itself, so that any improvement

will result in the loss of feasibility. All of these extreme points are nondominated. Point *E* is not nondominated, as keeping to the left in its improvement cone is still feasible, and all these points are better than *E*.

It can now be proved that the nondominated set is connected. This means that if two neighboring extreme points are nondominated, then all points (on the border of the feasible set) between them are also nondominated. This is the *nondominated frontier*, which is shown in bold in Figure 3.5. All points on this frontier are of interest to the decision maker, whose task is now to determine which of these solutions to choose. The choice will depend on criteria other than those already included in the model.

3.2 Solution Approaches to Vector Optimization Problems

This section will examine techniques that can be used to approximate the efficient frontier. While it is possible to use a modified simplex method to determine all extreme points on the efficient frontier, this is not only a lengthy process, but also something that leaves the decision maker with tons of solutions to manually compare. This is clearly not feasible. As a result, analysts typically determine a few solutions, and, based on the decision maker's response to those, will generate more solutions that attempt to reflect the decision maker's comments.

Two methods for this purpose stand out. One is the *weighting method*, and the other is called the *constraint method*. The basic idea of the weighting method is to first assign weights $w_1, w_2, \ldots, w_p$ to the p given objectives, and then aggregate them into a single new *composite objective*. The result is then a linear programming problem that can easily be solved. The optimal solution to this linear programming problem can then shown to be a point on the nondominated frontier, at least as long as the weights are positive.

As a numerical illustration, consider the example introduced in the previous section. Recall that the two objectives were Max $z_1 = 3x_1 + x_2$ and Max $z_2 = -2x_1 + x_2$. Suppose that we choose $w_1 = 5$ and $w_2 = 1$. This means that one unit of whatever the first objective measures is considered five times as important than one unit of what the second objective measures. Using these weights, the composite objective is then Max $z = w_1z_1 + w_2z_2 = 5(3x_1 + x_2) + 1(-2x_1 + x_2) = 13x_1 + 6x_2$. Using this objective in conjunction with the constraints of the problems results in the optimal solution $\bar{x}_1 = 5$, $\bar{x}_2 = 1$, with an objective value of $\bar{z} = 71$. For our purposes, the value of the aggregated objective function is irrelevant, it is more useful to take the solution obtained in the optimization and insert the optimal values into the two individual objective functions, resulting in $\bar{z}_1 = 16$ and $\bar{z}_2 = -9$. As can be verified in Figure 3.5, the solution $\bar{\mathbf{x}} = (5,1)$ is a point on the nondominated frontier. Table 3.1 provides a listing of different

weight combinations and the nondominated points they generate. Note that all nondominated solutions that this technique generates are at extreme points.

Table 3.1: Nondominated solutions generated by the weighting method

(w_1, w_2)	$z = w_1z_1 + w_2z_2$	$(\bar{x}_1, \bar{x}_2)$	$(\bar{z}_1, \bar{z}_2)$
(5, 1)	$13x_1 + 6x_2$	$D = (5, 1)$	(16, –9)
(3, 1)	$7x_1 + 4x_2$	$D = (5, 1)$	(16, –9)
(1, 1)	$x_1 + 2x_2$	$C = (2, 4)$	(10, 0)
(1, 3)	$–3x_1 + 4x_2$	$B = (1, 4)$	(7, 2)
(1, 5)	$–7x_1 + 6x_2$	$A = (0, 3)$	(3, 3)

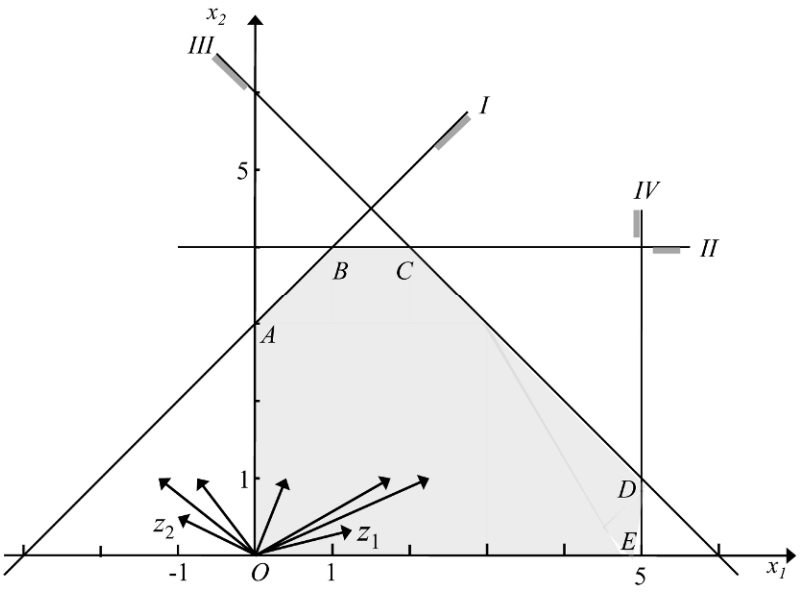


Figure 3.6

A graphical representation for our numerical problem is shown in Figure 3.6. Notice the objectives z_1 and z_2 as limits, the gradients of all of their combinations are between them. In particular, the gradients shown here are z_1, those generated by the weight combinations (5, 1), (3, 1), (1, 1), (1, 3), (1, 5), and z_2, in counterclockwise direction.

The second approach to generate nondominated solutions is the constraint method. It can be described as a technique that keeps one of the objective functions, while using the others as constraints with variable right-hand side values. This is done by first designating one of the p objectives as objective, while all others are

reformulated as constraints. Each objective of the type "Max z_k" will be rewritten as a constraint $z_k \geq b_k$ with a yet-to-be-determined value of b_k, while each original objective of the type "Min z_k" will be rewritten as a constraint of the type $z_k \leq b_k$ with variable values of b_k. The resulting linear programming can then easily be solved. The solution process is repeated for a number of combinations of values b_k, k=1, …, p. All solutions that are generated in this fashion are dominated. However, these solutions are not necessarily extreme points of the feasible set.

As an illustration, consider again the example introduced in the previous section. We will (arbitrarily) retain the first objective as an objective and reformulate the second objective as a constraint. The problem can then be written as

$$
\begin{aligned}
&\text{Max } z_1 = 3x_1 + x_2 \\
&\text{s.t.} \quad -x_1 + x_2 \leq 3 \\
&\qquad\quad x_2 \leq 4 \\
&\qquad x_1 + x_2 \leq 6 \\
&\qquad x_1 \leq 5 \\
&\qquad -2x_1 + x_2 \geq b_2 \\
&\qquad x_1, \quad x_2 \geq 0.
\end{aligned}
$$

At this point, we can solve the problem for a variety of values of b_2. A summary of solutions for some chosen values is displayed in Table 3.2.

Table 3.2: Nondominated solutions determined with the constraint method

b_2	$(\bar{x}_1, \bar{x}_2)$	$\bar{z}_1$
5	no feasible solution	
0	(2, 4)	10
–1	(2⅓, 3⅔)	10⅔
–2	(2⅔, 3⅓)	11⅓
–3	(3, 3)	12
–4	(3⅓, 2⅔)	12⅔
–5	(3⅔, 2⅓)	13⅓
–6	(4, 2)	14
–7	(4⅓, 1⅔)	14⅔
–8	(4⅔, 1⅓)	15⅓
–9	(5, 1)	16
–10	(5, 1)	16

For $b_2 = 0$, we obtain point C in Figure 3.5, and for $b_2 = -1, -2, \ldots, -8$, points between C and D are found, while $b_2 = -9$ and $b_2 = -10$ result in point D. Note that this selection of b_2 values misses the nondominated solutions A and B. This is why both, the weighting method and the constraint methods, are called approximation methods. As a matter of fact, using $b_2 = 1$ results in the solution $(\bar{x}_1, \bar{x}_2) = (1½, 4)$, which is the point halfway between B and C, a value of $b_2 = 2$

generates the solution $(\bar{x}_1, \bar{x}_2) = (1, 4)$, which is point B, a value of $b_2 = 3$ results in a solution $(\bar{x}_1, \bar{x}_2) = (0, 3)$, which is point A, and for values in excess of $b_4 = 4$, the problem has no feasible solutions. Figure 3.7 shows the feasible set of the problem and the constraints based on the second objective for various values of b_2. The bold points are the resulting optimal solutions.

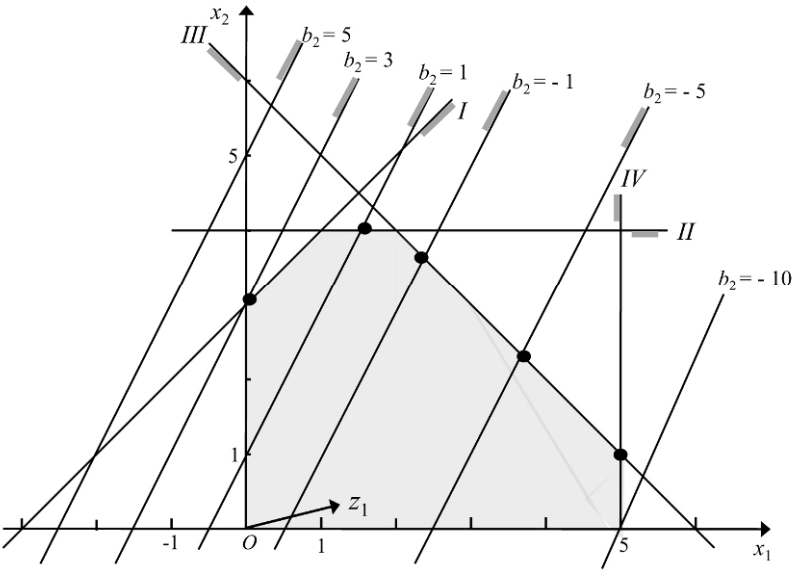


Figure 3.7

3.3 Goal Programming

When introducing goal programming, it is useful to return to the basic discussion in the first chapters of this book about constraints and objective functions. Recall that constraints express requirements that *must* be satisfied, while objective functions are for requirements that *should* be satisfied, if possible. While this distinction appears clear, the difference between "required" and "desired" is blurred in reality. Consider a simple budget constraint that expresses the condition that we cannot spend more than we have. While we should not do that, we could by borrowing money. All offices must fit into the space that we own—but we can rent some more. Don't use more employees than are available—but that's what temp agencies are for. Payments are due on a specific date—but we may be able to defer them. All of this is introduced into the discussion to demonstrate that many requirements are much softer than they appear. And this is why modelers should take precautions before formulating constraints, as they are absolute: if they cannot be satisfied by the given data, the solver will return a message indicating that there is no feasible solution.

Goal programming is one tool that attempts to deal with "soft constraints." The general idea was developed by the later Nobel laureate Simon in 1957, who introduced the concept of *satisficing*, a composite word that joins the concepts of "satisfy" and "suffice." The concept as applied to goal programming works as follows. The soft constraint is first formulated as a regular constraint. It is then reformulated as a goal constraint with the help of a *target value* (or *aspiration level*) and *deviational variables*. The target value is a number that expresses how many resources we have, what output we would like to achieve, or similar measures. We then introduce deviational variables d_ℓ^+ and d_ℓ^- which measure over- and underachievements, respectively. The easiest way to formulate goal constraints is to first write the requirement as a regular constraint, and then reformulate it as shown in Table 3.3.

Table 3.3: Formulation of soft constraints as goal constraints

Desired situation	Formulation of goal constraint	Contribution to the objective function
$LHS \leq RHS$	$LHS + d_\ell^- - d_\ell^+ = RHS$ $d_\ell^-, d_\ell^+ \geq 0$	Min d_ℓ^+
$LHS = RHS$		Min $d_\ell^- + d_\ell^+$
$LHS \geq RHS$		Min d_ℓ^-

The deviational variable d_ℓ^- is similar to a slack variable, and the deviational variable d_ℓ^+ resembles an excess variable. While it appears counterintuitive that slack and excess variables should appear in the same constraint, it really is not. As an example, consider a budget constraint that states that the actual expenditures should not exceed the available amount. First of all, the appearance of slack and surplus in this constraint indicates that the actual expenditures may be smaller than the available budget (underspending) or may exceed the actual budget (overspending). Secondly, there are technical reasons that ensure that at most one of the deviational variables can be positive, so that we cannot have over- and underspending at the same time.

As a numerical example, suppose that we have \$100 that we can spend on two items, food and entertainment (panem et circensis, as the Romans would have it). The amounts spent on the two items are x_1 and x_2, respectively, and the budget constraint would be $x_1 + x_2 \leq 100$. Reformulating it as a goal constraint leads to $x_1 + x_2 + d_\ell^- - d_\ell^+ = 100$. Suppose now that we have decided to spend \$60 of food and \$30 on entertainment. This means that the goal constraint is then $90 + d_\ell^- - d_\ell^+ = 100$ or $d_\ell^- - d_\ell^+ = 10$. Given that all variables, including the deviational variables, must satisfy the nonnegativity constraints, this implies that $d_\ell^- = 10$.

The meaning is that the present budget is "underachieved" by \$10, or, in more standard terms, there are \$10 left over.

If, on the other hand, we spend a total of, say, \$90 on food and \$30 on entertainment, the goal constraint reads 120+ $d_\ell^- - d_\ell^+ = 100$ or $d_\ell^- - d_\ell^+ = -20$. Again, given the nonnegativity of the deviational variables, the result is that $d_\ell^+ = 20$, indicating an "overachievement" (or, similarly, overspending) of the budget by \$20.

The last column in Table 3.3 then indicates the contribution to the overall objective made by the deviational variables introduced in a goal constraint. In the aforementioned budget constraint, the relation "actual amount spent ≤ amount available" was desired, so that goal programming, after rewriting the constraint as a goal constraint, will minimize the overachievement, i.e., overspending. Note that this approach does justice to the "softness" of the budget constraint by allowing overspending, but trying to minimize it. In practice, absolute or rigid constraints and soft constraints can often be distinguished by the way the requirements are worded. A telltale sign is the expression "if possible." Whenever it is appended to a requirement, it clearly indicates that formulation as a goal constraint is in order.

The next issue is then how to aggregate the deviational constraints into a single objective function. The original version of goal programming has multiple levels, each of them assumed to be infinitely more important than the next. This structure has been criticized profusely in the literature, even though the principle is common, even in linear programming: the absolute constraints are infinitely more important than the objective. This can be seen that if a constraint cannot be satisfied, we will obtain the signal "there exists no feasible solution" from the solver, regardless how good the objective function value is or can be.

In this book, we will restrict ourselves to a single level, on which we aggregate the deviational variables similar to the way we aggregated objective functions in the weighting method in the previous section of this chapter. The problem with such a procedure is commensurability. In other words, if one deviational variable expresses the overexpenditure of the budget (measured in dollars), while another expresses the underuse of manpower (measured in the number of employees), we cannot simply add these two together. When using weights, then these weights must include a tradeoff between the units. In the aforementioned example, a weight that is multiplies by the overexpenditure of the budget will have to express the importance of one dollar overexpenditure in relation to the underuse of one employee.

In order to illustrate the modeling process, consider the following numerical

Example: The owner of a chain of jewelry stores has to decide how to distribute parts of a new shipment of diamonds to five stores in a region. The first three stores of the chain are located in shopping malls. The following conditions have to be observed.

Absolutely necessary:
(a) Allocate between 1,000 and 1,200 carats to the five stores.
(b) Store 5 must receive at least 300 carats of diamonds.

Desired properties of the allocation:
(c) The stores in the malls should receive at least 80% of all the diamonds, if possible.
(d) The allocations to the stores in the mall should be equal to each other, if possible.
(e) The probabilities of theft in the stores have been estimated to be 0.1%, 0.1%, 9%, 2% and 3%. The owner would like to minimize the expected loss.

Requirement (e) takes priority in the list and is considered to be 25 times as important as requirement (d), which, in turn, is considered twice as important as (c).

In order to formulate the problem, we first define decision variables $x_1, x_2, \ldots, x_5$ as the quantity of diamonds allocated to stores 1, 2, 3, 4, and 5, respectively. The absolute constraints can then be written as

$$x_1 + x_2 + x_3 + x_4 + x_5 \geq 1{,}000 \quad (1)$$
$$x_1 + x_2 + x_3 + x_4 + x_5 \leq 1{,}200 \quad (2)$$
$$x_5 \geq 300. \quad (3)$$

Consider now requirement (c). Written as a constraint, the requirement can be formulated as $x_1 + x_2 + x_3 \geq 0.8(x_1 + x_2 + x_3 + x_4 + x_5)$ or, equivalently, as

$$0.2x_1 + 0.2x_2 + 0.2x_3 - 0.8x_4 - 0.8x_5 \geq 0.$$

Rewriting the requirement as a goal constraint with deviational variables, we obtain

$$0.2x_1 + 0.2x_2 + 0.2x_3 - 0.8x_4 - 0.8x_5 + d_1^- - d_1^+ = 0 \quad (4)$$

with Min d_1^- as the contribution to the objective function.

Next consider requirement (d). The average allocation to a store in the mall is $\frac{1}{3}(x_1 + x_2 + x_3)$, so that we would like to see $x_1 = \frac{1}{3}(x_1 + x_2 + x_3)$, $x_2 = \frac{1}{3}(x_1 + x_2 + x_3)$, and $x_3 = \frac{1}{3}(x_1 + x_2 + x_3)$. Rewriting the first of these constraints results in $\frac{2}{3}x_1 - \frac{1}{3}x_2 - \frac{1}{3}x_3 = 0$. As a goal constraint, we write

$$\tfrac{2}{3}x_1 - \tfrac{1}{3}x_2 - \tfrac{1}{3}x_3 + d_2^- - d_2^+ = 0 \tag{5}$$

With the objective function contribution Min $d_2^- + d_2^+$. Similarly, we obtain the goal constraints

$$-\tfrac{1}{3}x_1 + \tfrac{2}{3}x_2 - \tfrac{1}{3}x_3 + d_3^- - d_3^+ = 0 \tag{6}$$

and

$$-\tfrac{1}{3}x_1 - \tfrac{1}{3}x_2 + \tfrac{2}{3}x_3 + d_4^- - d_4^+ = 0 \tag{7}$$

with the objective function contributions Min $d_3^- + d_3^+$ and Min $d_4^- + d_4^+$, respectively.

Finally, consider requirement (e). The original objective is written as

$$\text{Min } z = 0.001x_1 + 0.001x_2 + 0.09x_3 + 0.02x_4 + 0.03x_5.$$

Setting the expected loss at some unattainably low level, e.g., $z = 0$, we can then require that the expected loss is no larger than that level, if possible (which it is not, so that we minimize the overachievement). This is then written as

$$0.001x_1 + 0.001x_2 + 0.09x_3 + 0.02x_4 + 0.03x_5 + d_5^- - d_5^+ = 0 \tag{8}$$

with the objective function contribution Min d_5^+.

The problem can then be written as

$$\text{Min } z = 50\,d_5^+ + 2(d_2^- + d_2^+) + 2(d_3^- + d_3^+) + 2(d_4^- + d_4^+) + d_1^-$$

s.t. constraints (1) – (8) and the nonnegativity constraints for all variables.

The optimal solution allocates 466.69 carats of diamonds to store 1, another 233.31 carats of diamonds to store 2, no diamonds to stores 3 and 4, and the minimally required 300 carats to store 5. The expected loss is measured by the overachievement d_5^+, whose optimal value is 9.7. Due to the overriding importance of the expected loss, equality among the mall stores is no longer achieved.

Exercises

Problem 1 (improvement cone): Consider the following two vector optimization problems (to simplify matters, the constraints have been ignored):

(a) P_1: Max $z_1 = 5x_1 + 2x_2$, Max $z_2 = -2x_1 - 2x_2$, Max $z_3 = 3x_1$, and

(b) P_2: Max $z_1 = x_2$, Max $z_2 = 4x_1 - x_2$, Min $z_3 = x_1 + x_2$.

Plot each of these two problems individually and determine the improvement cone.

Solution: The improvement cone for (a) is shown in Figure 3.8a, the improvement cone for (b) is empty as shown in Figure 3.8b.

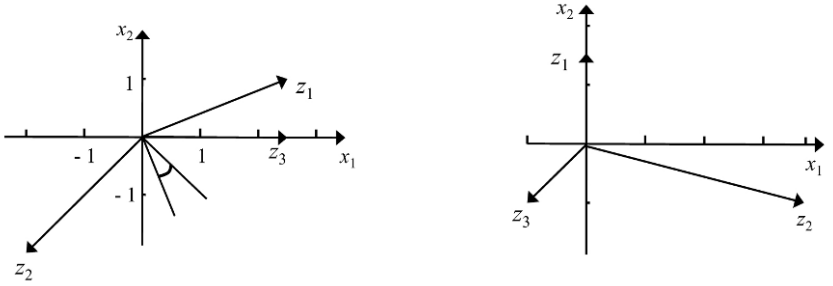


Figure 3.8

Problem 2 (nondominated frontier and composite objective, graphical): Consider the following linear programming problem:

$$
\begin{aligned}
&\text{P: Max } z_1 = x_1 + 2x_2 \\
&\text{s.t.} \quad -2x_1 + x_2 \le 2 \\
&\qquad\quad x_1 + x_2 \le 5 \\
&\qquad\quad x_1 \le 3 \\
&\qquad\quad x_1, \; x_2 \ge 0.
\end{aligned}
$$

(a) Plot the constraints and determine the feasible set.
(b) Graph the gradient of the objective function and use the graphical solution technique to determine the optimal point. Compute the exact coordinates of the optimal point and its value of the objective function.
(c) Consider a second objective function Max $z_2 = 2x_1 - x_2$. Ignoring the first objective, what is the optimal point? Compute its exact coordinates and its value of the objective function.

(d) Determine the nondominated frontier given the two objectives.
(e) Use the two objectives above to construct the composite objective function with weights $w_1 = ¾$ and $w_2 = ¼$. What is the optimal solution with this objective?

Solution: (a) The solutions are based on Figure 3.9.

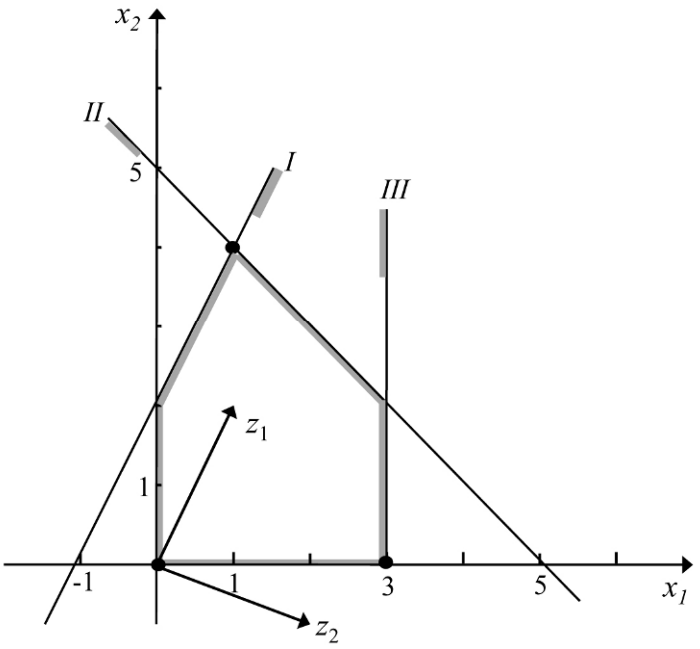


Figure 3.9

(b) The exact coordinates of the optimal solution are $\bar{\mathbf{x}} = (1, 4)$ with value of the objective function $\bar{z}_1 = 9$.
(c) The optimal solution with the second objective is $\bar{\mathbf{x}} = (3, 0)$ with $\bar{z}_2 = 6$.
(d) Shown by the bold line in Figure 3.10.
(e) Shown in Figure 3.10.

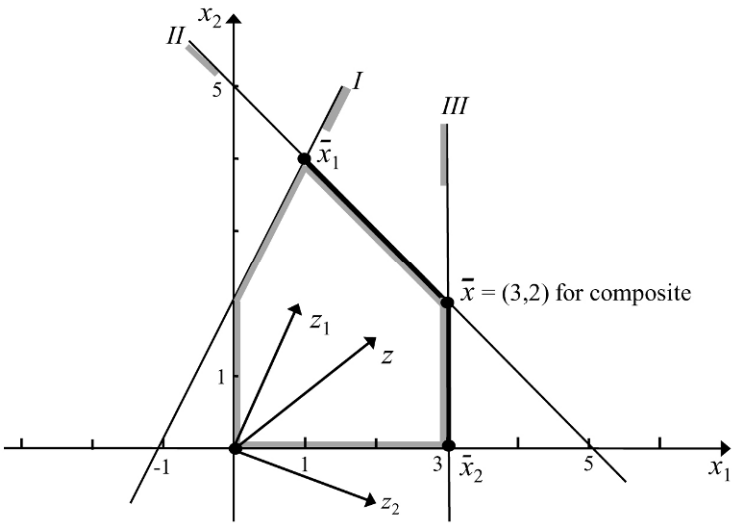


Figure 3.10

Problem 3 (vector optimization, nondominated frontier): Consider the following vector optimization problem:

$$\begin{aligned} \text{P: Min } z_1 &= x_1 + x_2 \\ \text{Max } z_2 &= 2x_1 + x_2 \end{aligned}$$

$$\begin{aligned} \text{s.t. } x_1 &\leq 3 \\ x_2 &\leq 2 \\ -x_1 + x_2 &\leq 1 \\ x_1, \quad x_2 &\geq 0. \end{aligned}$$

(a) Graph the constraints, clearly indicate the feasible set, and graph the directions of both objective functions.
(b) Determine the improvement cone in a separate graph.
(c) Plot the improvement cone at each extreme point of the feasible set and determine the efficient frontier. Clearly describe the efficient frontier.

Solution:

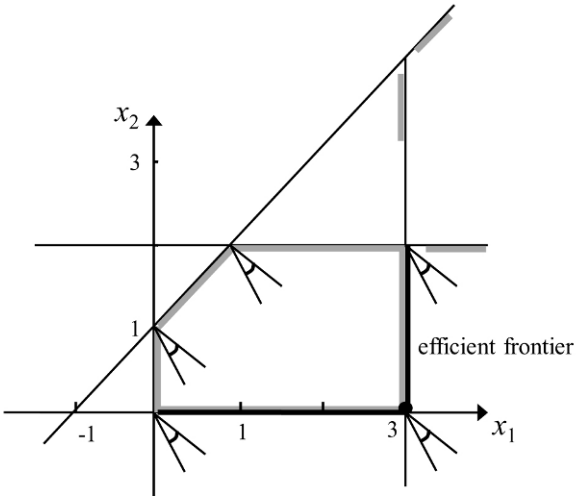


Figure 3.11

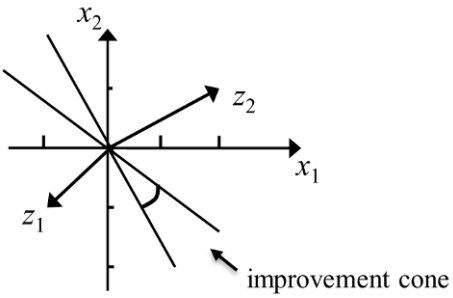


Figure 3.12

Problem 4 (weighting method): Consider again the vector optimization problem in Problem 3.

(a) Use the weighting method with weight combinations **w** = (5, 1), (3, 1), (1, 1), (1, 3), and (1, 5) to determine nondominated solutions.

(b) Use the constraint method by keeping the first objective and using the second objective as constraint with a variable right-hand side value b_2.

(c) Repeat (b) by keeping the second objective and using the first objective as a constraint with variable right-hand side b_1.

Solution:

(a) The different weight combinations result in the solutions shown below.

w	$\bar{\mathbf{x}}$
5, 1	0, 0
3, 1	0, 0
1, 1	3, 0
1, 3	3, 2
1, 5	3, 2

(b) The method generates the noninferior solutions shown below for various values of b_2 in the constraint $2x_1 + x_2 \leq b_2$.

b_2	$\bar{\mathbf{x}}$	$\bar{z}_1$
–10	0, 0	0
0	0, 0	0
5	2.5, 0	2.5
6	3, 0	3
7	3, 1	4
8	3, 2	5
9	no feasible solution	

(c) The method generates the noninferior solutions shown below for various values of b_2 in the constraint $x_1 + x_2 \geq b_1$.

b_1	$\bar{\mathbf{x}}$	$\bar{z}_2$
10	3, 2	8
5	3, 2	8
4	3, 1	7
3	3, 0	6
2	2, 0	4
1	1, 0	2
0	0, 0	0
–1	no feasible solution	

Problem 5 (goal programming formulation): A product P is to be blended from three ingredients I_1, I_2, and I_3. Firm requirements dictate that at least 100 lbs of P are to be blended, and that the average cost of the blend per pound do not exceed \$2.80, given that one pound of the three ingredients costs \$5, \$3, and \$2 for I_1, I_2, and I_3, respectively. In addition, it would be desirable if 20% of P were to be I_1. Similarly, the decision maker would like to have P consist of no more than 50% of the cheap ingredient I_3, if possible. This desirable feature is about half as important as the former desirable feature. Formulate as a goal programming problem.

Solution: As there is only a single product, we only need variables with a single subscript. Define variables x_1, x_2, and x_3 as the quantities of the three respective ingredients in the product. The constraints can then be written as

$$x_1 + x_2 + x_3 \geq 100 \text{ and}$$
$$5x_1 + 3x_2 + 2x_3 \leq 2.8(x_1 + x_2 + x_3).$$

The first of the two desirable properties, written as a constraint is

$x_1 \geq 0.2(x_1 + x_2 + x_3)$. Rewriting as a goal constraint, we obtain

$$x_1 + d_1^- - d_1^+ = .2(x_1 + x_2 + x_3)$$

with objective function contribution Min d_1^-. The latter desirable property, written as a constraint, is

$x_3 \leq .5(x_1 + x_2 + x_3)$. Rewritten as a goal constraint, we obtain

$$x_3 + d_2^- - d_2^+ = .5(x_1 + x_2 + x_3)$$

with objective function contribution Min d_2^+. The objective function is

$$\text{Min } z = 2\, d_1^- + d_2^+ .$$

Solving the problem results in 15, 35, and 50 lbs of the three ingredients being used, so that exactly 100 lbs are blended, whose price is exactly equal to the required value of 2.8. It is apparent that the product includes 15% of I_1, 5% short of the desired target. On the other hand, the upper limit of 50% of I_3 is satisfied as an equation.

4 Integer Programming

Not too long after more and more applications of linear programming were developed, it became apparent that in some of these applications, the variables would not be able to attain just any (nonnegative) value, but should be integers. As a simple applications, if a variable has been defined to denote the number of cans of beans manufactured in the planning period, then surely it would make no sense to make, say, 1,305,557.3 cans: the last 0.3 cans would have to be rounded up or down. While this may be an acceptable practice when dealing with this application (after all, it makes very little difference whether or not we make 0.3 cans more or less), in other applications this may make a huge difference. For instance, assigning airplanes to routes or trucks to deliveries may very well make the difference between gain and loss. Furthermore, simply rounding up or down a noninteger (usually referred to as a continuous solution) will not necessarily result in an optimal integer solution. We will demonstrate this fact below.

Even though the difference may be blurry, it may be useful to distinguish between variables that are naturally required to be integer (such as the number of trucks to be used for deliveries, the number of work crews dispatched to a construction site, or the number of drums of hazardous material shipped from one site to another), and so-called logical variables that also must be integer and are introduced for logical reasons. Below, we will provide a number of examples of the latter.

Integer programming problems were first discussed by Gomory in the 1950s, who also devised a solution technique for them. Gomory's class of techniques is called *cutting plane techniques*, and we will describe their basic idea below. A breakthrough is the 1961 contribution by Land and Doig, whose *branch-and-bound method* remains the standard solution technique to this day.

4.1 Definitions and Basic Concepts

In order to define integer programming problem, we will start with a standard linear programming problem. As an example, consider the problem

H.A. Eiselt and C.-L. Sandblom, *Operations Research: A Model-Based Approach*,
DOI 10.1007/978-3-642-10326-1_4, © Springer-Verlag Berlin Heidelberg 2010

$$
\begin{aligned}
P_1\text{: Max } z &= x_1 + x_2 \\
\text{s.t.} \quad & 3x_1 + 5x_2 \le 15 \\
& 5x_1 + 2x_2 \le 10 \\
& x_1, \quad x_2 \ge 0.
\end{aligned}
$$

The only difference between the standard linear programming problem above and an integer programming problem is that some or all of the variables are, in addition to be required to be nonnegative, are *also* required to be integer. It is very helpful to think of the integrality condition as an *additional* requirement. All integer programming problems that require some, but not all, variables, to be integer, are called *mixed-integer linear programming problems* (or *MILP*s), while problems, in which all variables are required to assume integer values, are called *all-integer linear programming* (or *AILP*) problems.

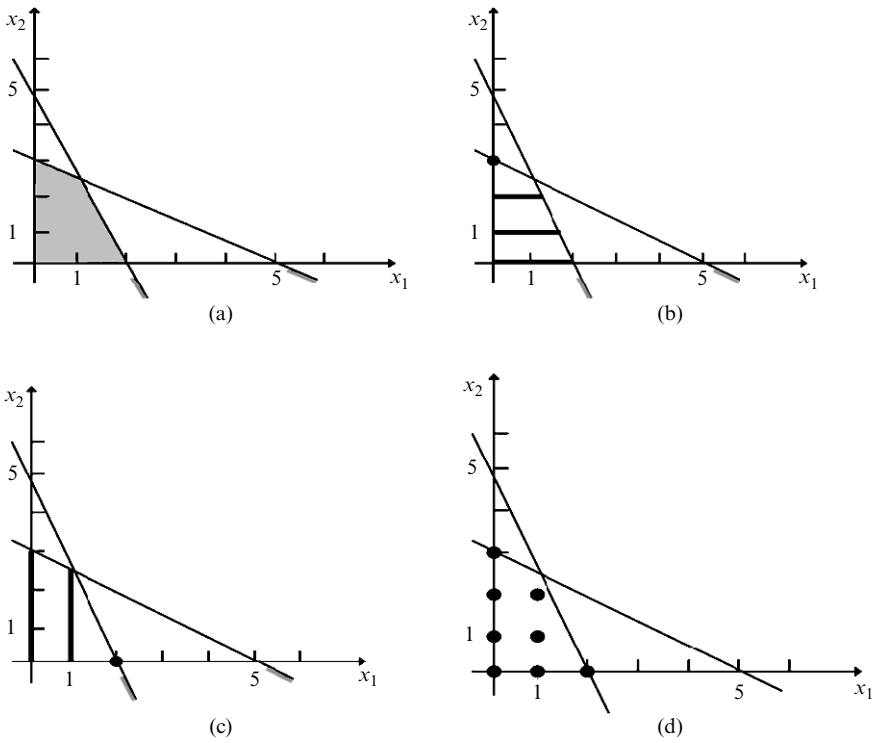
If we replace in the above example the two nonnegativity constraints by $x_1 \ge 0$, and $x_2 \ge 0$ *and* integer, then we have a mixed integer programming problem, which we may call P_2. Similarly, if we replace the nonnegativity constraints in P_1 by the conditions $x_1 \ge 0$ *and* integer and $x_2 \ge 0$, we have another mixed-integer linear programming problem P_3. Finally, if we replace in P_1 both nonnegativity constraints by $x_1 \ge 0$ *and* integer as well as $x_2 \ge 0$ *and* integer, we then have the all-integer linear programming problem P_4.

The graphical representations of the feasible sets of the four problems P_1, P_2, P_3, and P_4 are shown in Figures 4.1a – 4.1d. Figure 4.1a shows the usual linear programming problem with the feasible set shaded. The mixed-integer linear programming problem P_2 is shown in Figure 4.1b; here, only the bold horizontal bars are feasible, as only those guarantee that the variable x_2 is integer, while in addition, respecting the given constraints. Similarly, only the bold vertical bars in Figure 4.1c are feasible for the mixed-integer linear programming problem P_3. Finally, the feasible set of the all-integer linear programming problem P_4 consists exclusively of the grid points shown in Figure 4.1d. Clearly, the feasible set of this problem is smallest, as it is the most constrained. Incidentally, the optimal solutions of the four problems are as follows:

P_1: $\bar{\mathbf{x}} = (1.0526, 2.3684)$ with $\bar{z} = 3.4211$,
P_2: $\bar{\mathbf{x}} = (1, 2.4)$ with $\bar{z} = 3.4$,
P_3: $\bar{\mathbf{x}} = (1.2, 2)$ with $\bar{z} = 3.2$, and
P_4: $\bar{\mathbf{x}} = (1, 2)$ with $\bar{z} = 3$.

Again, it is apparent that moving from the least restricted problem P_1 on to the more restricted problems P_2 and P_3 to the most restricted problem P_4, the values of the objective function are getting worse, i.e., they decrease in maximization problems, while they would increase in minimization problems.

Furthermore, it is important to realize that none of the integer programming problems has its optimal solution at an extreme point, which is always the case in linear programming (recall Dantzig's corner point theorem).



Figures 4.1a – d

In the above example it appears as if integer programming problems could be solved by first solving the (simpler) linear programming problem and then rounding the solution up or down. This is, however, not the case. As an example, consider the following all-integer linear programming problem P_5:

$$
\begin{aligned}
P_5\text{: Max } z &= 2x_1 + x_2 \\
\text{s.t.} \quad & 7x_1 + 48x_2 \leq 84 \\
& -x_1 + 12x_2 \geq 3 \\
& x_1, \quad x_2 \geq 0 \text{ and integer.}
\end{aligned}
$$

The two solid lines in Figure 4.2 show the two constraints (ignore the broken line for now), and the bold dots indicate the all-integer solutions. The optimal solution of the linear programming problem without the integrality requirements is $\bar{\mathbf{x}} = (6.5455, 0.7955)$ with $\bar{z} = 13.8864$, while the optimal solution of the all-integer programming problem is $\bar{\mathbf{x}} = (5, 1)$ with $\bar{z} = 11$. This is a solution that cannot be obtained by simple rounding.

Worse yet, if we were to change the right-hand side value of the second constraint from "3" to "13," one of the constraints in Figure 4.2 moves in parallel fashion to the broken line. The feasible set for the linear programming problem without integrality conditions is the triangle with the vertices (0, $1\frac{1}{12}$), (0, 1¾), and the point marked with $\bar{x}'_{LP}$, which has coordinates (2.9091, 1.3258) and a value of the objective function of 7.1439. The all-integer linear programming problem does, however, hove no feasible solution (notice that there are no bold dots in the triangle described above).

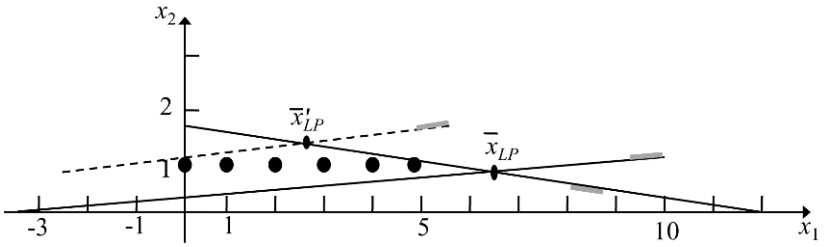


Figure 4.2

Finally, consider the feasible set generated by the following constraints:

$$\begin{aligned} x_1 + \ x_2 &\le 10 \\ 5x_1 + 3x_2 &\ge 15 \\ x_1 \qquad &\le \ 6 \\ x_2 &\le \ 7 \\ x_1, \quad x_2 &\ge \ 0. \end{aligned}$$

The feasible set is shown in Figure 4.3.

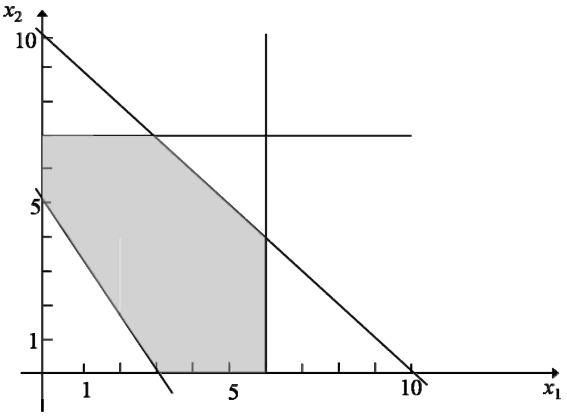


Figure 4.3

Inspection reveals that the feasible set has extreme points (3, 0), (6, 0), (6, 4), (3, 7), (0, 7), and (0, 5). In other words, all corner points of the feasible set are integer-valued. If we were to solve a linear programming problem with this feasible set and any objective function, the optimal solution would—as proved by Dantzig's "corner point theorem"—be one of these points, and thus integer without us actually requiring it. There are some large classes of linear programming formulations that fall into this category, i.e., resulting in integer solutions without explicitly including integer requirements. This is a very appealing feature, since linear programming problems are generally much easier to solve than their integer programming counterparts. We will return to these problems in the chapter on network models.

The remainder of this section will discuss the relations between the objective values of linear programming problems and integer linear programming problems. In order to facilitate the discussion, we need to introduce some terminology. Consider any mixed- or all-integer linear programming problem P. Its *linear programming relaxation* P_{rel} is the very same problem P except with all of the integrality conditions deleted. Suppose that P is a maximization problem and its optimal value of the objective function is $\bar{z}_{IP}$. Let now $\bar{z}_{LP}$ denote the optimal value of the objective function of its linear programming relaxation, then we find that

$$\bar{z}_{IP} \leq \bar{z}_{LP}.$$

The reason for this relation is easy to see. Starting with the relaxation, we obtain the integer programming problem by adding constraints, namely the integrality constraints. And whenever constraints are added to a problem, the value of the objective function gets worse, i.e., lower in case of maximization problems and

higher in case of minimization problems. And that is exactly what is captured by the above relation. Note that the inequality is reversed in case of minimization problems. In other words, the linear programming relaxation provides an upper bound on the value of the objective function of the integer programming problem in case of maximization problems, while it is a lower bound in case of minimization problems.

As a numerical example, consider the all-integer linear programming problem P_5 introduced earlier in this section. The value of the objective function of the all-integer solution was found to be $\bar{z}_{IP} = 11$, while the linear programming relaxation has an optimal objective value of $\bar{z}_{LP} = 13.8864$.

Not only does the numerical example satisfy the inequality shown above as expected, but it also allows us to define the gap between the objective values of an integer programming problem and its linear programming relaxation. Formally, the *absolute integrality gap* is defined as the difference $\bar{z}_{\mathrm{LP}} - \bar{z}_{\mathrm{IP}}$, while the *relative integrality gap* is defined as $\dfrac{\bar{z}_{LP} - \bar{z}_{IP}}{\max\{|\bar{z}_{LP}|, |\bar{z}_{IP}|\}}$. In the numerical example the absolute integrality gap is 13.8864 – 11 = 2.8864, while the relative integrality gap is 2.8864/13.8864 = .2079. The relative integrality gap is usually a good indicator of the degree of difficulty of the problem. Any problem with a relative integrality gap in excess of 0.1 or 10% is fairly difficult, while problems with integrality gaps in excess of 0.5 or 50% are typically really difficult. The integrality gap can, of course, only be computed after the problem and its relaxations have been solved, thus diminishing its usefulness. However, often good approximations for $\bar{z}_{IP}$ and $\bar{z}_{LP}$ are known.

4.2 Applications of Integer Programming

This section will introduce some classes and principles of applications of integer programming problems. The simplest integer programming problem are the so-called *knapsack problems*. Formally, knapsack problems have an objective and a single constraint. The story behind the name can be told as follows. A backpacker wants to decide which items to take into the woods. Items to be considered include a tent, a sleeping bag, a stove, stove fuel, map, compass, and so forth. With each item, the backpacker associates two numbers: one that expresses the value of the item, and the other its weight. The problem is then to choose items, so as to maximize their total value to the backpacker, while the total weight should not exceed a prespecified limit.

There are a number of different version of knapsack problems. One version only allows the backpacker to either pack an item or leave it home, while another

version permits the hiker to take any (integer) number of units of an item. The definition of variables is similar regardless of the application. In case we can only decide to either pack an item or not, we will define a variable for item j, such that $y_j = 1$, if we include the item in our pack, and 0 otherwise. In contrast, if we are allowed to take any number of units of item j, we will definite variables that are defined as y_j: the number of items of type j that we include in our pack, so that $y_j \geq 0$ and integer. (As a matter of fact, in the former case we can also think of the variable in terms of the number of items taken, only that this number is restricted to no more than one). Note that in order to distinguish integer variables from those that do not have to satisfy integrality, we will use y_j for integer variables and x_j for continuous variables.

Knapsack problems occur in a variety of guises. The "cargo loading" prototype problem is of this variety. For continuous knapsack problems (i.e., those, in which the variables do not have to be integers or even zeroes and ones), see Exercise 2 in Section 2.5. Another popular example of a knapsack problem is *capital budgeting*. As a numerical illustration, suppose that a developer can engage in five different projects, *viz*., a highrise building, a shopping mall, an amusement park, a warehouse, and an airport. The expected profit contributions and resource consumption (e.g., the number of construction workers needed for the respective projects) are shown in Table 4.1.

Table 4.1: Profit contributions and resource consumption of projects

	Highrise	Shopping mall	Amusement park	Warehouses	Airport
Profit contribution (in \$1M)	10	6	12	2	7
Resource consumption	4	2	5	1	3

Assume now that the developer has seven work crews at his disposal, and suppose that it is not possible to hire additional work crews. We can then define variables $y_j = 1$, if the developer is to engage in project j, and 0 otherwise (clearly, engaging in partial projects is not feasible and none of the projects can be performed more than once), so that the problem can be formulated as follows:

$$\begin{aligned} \text{P: Max } z &= 10y_1 + 6y_2 + 12y_3 + 2y_4 + 7y_5 \\ \text{s.t.} \quad & 4y_1 + 2y_2 + 5y_3 + 1y_4 + 3y_5 \leq 7 \\ & y_1, \; y_2, \; y_3, \; y_4, \; y_5 = 0 \text{ or } 1. \end{aligned}$$

The linear programming relaxation that includes just the objective, the single constraint, and the nonnegativity constraints, has the optimal solution $\bar{y}_2 = 3.5$, $\bar{y}_1 = \bar{y}_3 = \bar{y}_4 = \bar{y}_5 = 0$ with the objective value $\bar{z} = 21$. Adding upper bounds $y_1 \leq 1$,

$y_2 \leq 1, y_3 \leq 1, y_4 \leq 1$, and $y_5 \leq 1$ results in $\bar{y}_1 = \bar{y}_2 = 1$, $\bar{y}_3 = 0.8$, and $\bar{y}_4 = \bar{y}_5 = 0$ with an objective value of $\bar{z} = 18.4$. Finally, requiring integrality of all variables, we obtain the optimal solution $\bar{y}_1 = \bar{y}_2 = \bar{y}_4 = 1$, $\bar{y}_3 = \bar{y}_5 = 0$ with the objective value $\bar{z} = 18$.

Many beginners think that knapsack problems—and particularly zero-one knapsack problems—are so simple that modern computational equipment must surely be able to solve such problems by simply enumerating all solutions, without having to resort to complicated algorithms. Nothing could be further from the truth. In order to demonstrate this, compute the number of different solutions of a zero-one integer programming problem. A problem with a single variable has two solutions, as the variable can assume the values of zero and one. With two variables, we have four solutions: (0, 0), (0, 1), (1, 0), and (1, 1). With three variables, we already have eight different solution, with four variables, there are sixteen different solutions, and so on. As a matter of fact, adding a single variable will double the number of solutions of the problem. The reason is this: the variable that we add can have a value of zero or one. If it were zero, together with all possible solution of the other variables, we have just as many solutions as before. The same applies if the new variable were equal to one, so that adding a variable doubles the number of solutions. This means that in case of n variables, we will have 2^n different solutions that have to be examined. This means that for $n = 10$, we have 1,024 solutions, for $n = 20$, there will be a million solutions, for $n = 30$ there is a billion, for $n = 40$ a trillion, and so forth. Clearly, even if a computer could examine a quadrillion solutions within a single second, (which is pretty much the limit by today's standards), a problem with 100 variables (a tiny problem by today's standards), would require more than 40 million years to examine all solutions. For business problems, that time frame appears excessive.

It is no wonder that many users resort to heuristic algorithms for the solution of integer programming problems. We will discuss exact and heuristic solution techniques in Section 3 of this chapter.

4.2.1 Cutting Stock Problems

The first integer application we will discuss is a model, in which the variables quite naturally must assume integer values. Cutting stock problems (or, alternatively, stock cutting or trim loss problems) are among the early applications of integer linear programming. The first studies concerned paper rolls, whose width is fixed, but which can be cut to desired lengths. The decision maker then has a number of larger rolls of paper of given length, which he has to cut down to smaller rolls that are in demand. This is what is called a one-dimensional problem, as only the length of the rolls is cut.

In order to explain the formulation, suppose that a home improvement store carries wooden rods in a standard profile and width. They presently have two lengths, 12 ft and 10 ft. In particular, they have twenty 12 ft rods and twenty-five

10 ft rods in their warehouse. Management anticipates a need for sixty 8 ft rods, forty 5 ft rods, and seventy-five 3 ft rods. In order to obtain the desired lengths, we can either cut existing rods at a cost of 50¢ per cut, or purchase new rods at a cost of \$2, \$1.50, and \$1.10 for the 8 ft, 5 ft, and 3 ft rods, respectively.

There are two common types of objectives. Management could either attempt to minimize the waste produced in the process, or could minimize the costs incurred in the process. Minimizing waste is a popular option, yet it is nontrivial from a conceptual point of view. A small piece, say a 2 ft rod, cannot be used and is considered a complete waste. A larger piece, say a 4 ft rod, however, while formally twice the waste of a 2 ft rod, can still be used as to satisfy some future demand, even if it is not used in this planning period. As a result, we will simply minimize the total cost incurred in the process.

In order to formulate the problem, it is mandatory that we first devise a cutting plan. A cutting plan will include all meaningful cutting patterns. Patterns that are undesirable, either because they produce too much waste, are too difficult to cut, or for some other reason, are simply not included in the cutting plan.

The cutting plan for this example is shown in Figure 4.4, where the meaningful cutting patters are numbered 1 to 8 from top to bottom.

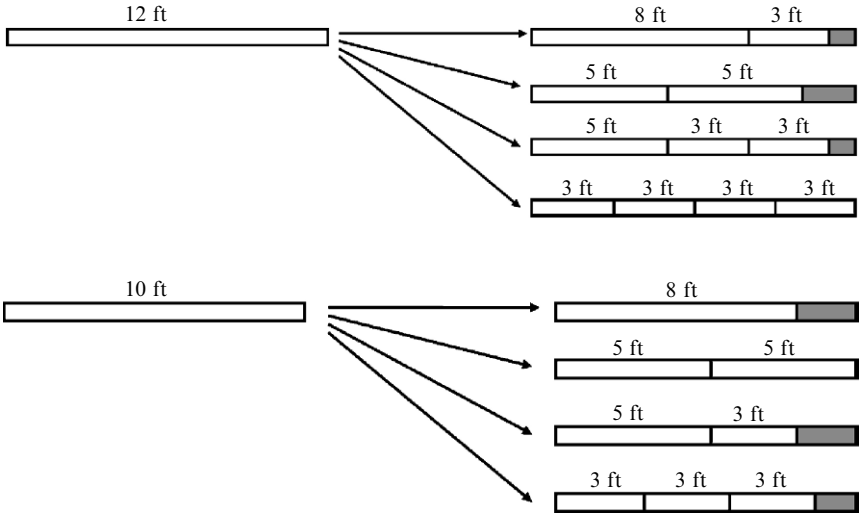


Figure 4.4

We can now define the decision variable y_1 as the number of times that pattern 1 is cut, and similarly for the remaining seven patterns. This takes care of the cutting we have to do. In addition, we also require variables v_1, v_2, and v_3 that determine the number of 8 ft, 5 ft, and 3 ft rods that we purchase in addition to cutting longer rods to the required sizes. Consequently, the objective function consists of two

major components, the cost of cutting and the cost of purchasing. Consider first the cutting costs. Pattern 1 requires two cuts, so that each time pattern 1 is cut, it will cost \$1. Given that pattern 1 is cut y_1 times, the cost contribution of the first pattern is $1y_1$. Similarly, pattern 8 requires three cuts or \$1.50 each time it is cut. Since pattern 8 is cut y_8 times, its cost contribution is $1.5y_8$. Adding the purchasing costs for the rods that are not cut, the objective function can then be formulated as

$$\text{Min } z = 1y_1 + 1y_2 + 1.5y_3 + 1.5y_4 + 0.5y_5 + 0.5y_6 + 1y_7 + 1.5y_8 + 2v_1 + 1.5v_2 + 1.1v_3.$$

As far as the constraints are concerned, there will be two types, *viz*., supply and demand constraints.

Consider first the supply constraints. In words, they state that the number of patterns cut from an existing length cannot exceed the number of rods that are available. Constraints of this type have to be formulated for each existing length. Consider first the 12 ft length. It is used in patterns 1, 2, 3, and 4, which, as we already know, are cut y_1, y_2, y_3, and y_4 times, respectively. Given that we have twenty 12 ft rods available, we can formulate the supply constraint for the 12 ft length as

$$y_1 + y_2 + y_3 + y_4 \leq 20. \tag{1}$$

Similarly, the supply constraint for the 10 ft rods is

$$y_5 + y_6 + y_7 + y_8 \leq 25. \tag{2}$$

The demand constraints are somewhat more difficult to formulate. As an example, consider the first required length of 8 ft. It is produced by patterns 1 and 5. Each time we cut pattern 1, we generate one 8 ft rod. Since we cut this pattern y_1 times, the number of 8 ft rods produced by cutting pattern 1 is $1y_1$. Similarly, since each time pattern 5 is cut, we generate a single 8 ft rod, we make a total of $1y_5$ 8 ft rods by cutting pattern 5. Since the only other way to obtain 8 ft rods is to purchase them (and we already have decided to buy v_1 of them), the total number of 8 ft rods that we will have is $1y_1 + 1y_5 + v_1$, a number that must be large enough to satisfy the demand of 60 units. The demand constraint for the 8 ft rods can thus be written as

$$y_1 + y_5 + v_1 \geq 60. \tag{3}$$

As far as 5 ft rods are concerned, the cutting plan reveals that they are generated by patterns 2, 3, 6 and 7. Since each time pattern 1 is cut, we produce two 5 ft rods, and as pattern 1 is cut y_1 times, we will produce a total of $2y_1$ 5 ft rods by cutting pattern 1. Applying similar arguments for the other three patterns that generate 5 ft rods, the constraint for these rods is

$$2y_2 + 1y_3 + 2y_6 + 1y_7 + v_2 \geq 40. \quad (4)$$

Finally, the 3 ft rods. They are generated by patterns 1, 3, 4, 7 and 8. Each time one of these patterns is cut, we generate 1, 2, 4, 1, and 3 of the required 3 ft rods. As a result, we can write the constraint for the 3 ft rods as

$$1y_1 + 2y_3 + 4y_4 + 1y_7 + 3y_8 + v_3 \geq 75. \quad (5)$$

The cutting stock problem can then be written as

$$\text{Min } z = 1y_1 + 1y_2 + 1.5y_3 + 1.5y_4 + 0.5y_5 + 0.5y_6 + 1y_7 + 1.5y_8 + 2v_1 + 1.5v_2 + 1.1v_3.$$

s.t. constraints (1) – (5)
$y_1, y_2, \ldots, y_8; v_1, v_2, v_3 \geq 0$ and integer.

Solving the problem results in $\bar{y}_1 = 2$, $\bar{y}_2 = 0$, $\bar{y}_3 = 0$, $\bar{y}_4 = 18$, $\bar{y}_5 = 5$, $\bar{y}_6 = 20$, $\bar{y}_7 = 0$, and $\bar{y}_8 = 0$, as well as $\bar{v}_1 = 53$, $\bar{v}_2 = 0$, and $\bar{v}_3 = 1$. This leaves none of the existing rods left over, and the demand is exactly satisfied.

Solving the same problems with demands of 20, 15, and 18 for the 8 ft, 5 ft, and 3 ft rods results in $\bar{y}_1 = 6$, $\bar{y}_2 = 0$, $\bar{y}_3 = 0$, $\bar{y}_4 = 3$, $\bar{y}_5 = 14$, $\bar{y}_6 = 8$, $\bar{y}_7 = 0$, and $\bar{y}_8 = 8$, as well as $\bar{v}_1 = 0$, $\bar{v}_2 = 0$, and $\bar{v}_3 = 0$. In this case, nothing is purchased, we have eleven 12 ft rods and three of the existing 10 ft rods left over, and the demand is exactly satisfied for the 8 ft and 3 ft rods, while one 5 ft rod is cut but not used.

As expected, things get more complicated when cutting is possible in two dimensions. However, it is not the formulation that becomes more difficult, but the cutting plan that may now include many patterns. Just imagine cutting an irregular piece of fabric that may be shifted and tilted by infinitesimally small amounts to any of its sides, resulting in infinitely many patterns. To simplify matters, assume that the patterns are regular. As a numerical example, consider a single board of plywood of size 5 ft by 8 ft, and assume that we need two types of boards, type T_1 is of size 2 ft × 4 ft, and type T_2, which measures 3 ft square. Two of the many possible patterns are then shown in Figure 4.5, where the shaded parts indicate waste.

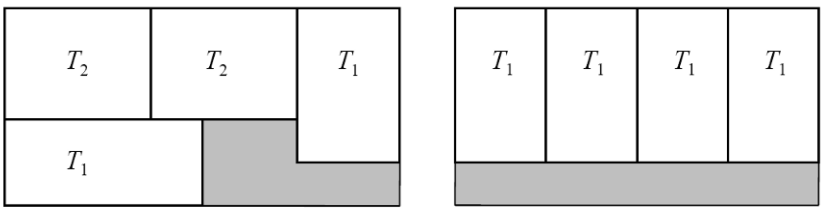


Figure 4.5

When evaluating these patterns, we could resort to the amount of waste that is generated. The first pattern uses 34 sq ft out of the given 40 sq ft for a usage rate of 85%. Pattern 2, in contrast, uses only 32 sq ft for a usage rate of 80%, so it appears that pattern 1 is more efficient. This is, however, not necessarily the case. Pattern 2 is easy to cut: Adjust the saw once to a 1 ft width, cut off the shaded part at the bottom, readjust the machine to a 2 ft cutting width, and continue to cut the four desired T_1 pieces. Things are much more complicated when cutting pattern 1. First of all, we should point out that only so-called *guillotine cuts* are feasible. These are cuts that the existing piece all the way through. This restriction is important, as non-guillotine cuts will result in many operator errors. Probably the best way to cut pattern 1 is to first adjust the machine to a 2 ft width, cut the T_1 piece on the right with the piece of waste at the bottom, then turn the remaining board and cut the T_1 piece at the bottom (this way, we do not have to readjust the machine all the time), then readjust the machine to a 3 ft cutting width and cut the top left piece in the middle, generating the two T_2 pieces. And then we have to readjust the machine again to cut off the waste at the bottom of the two T_1 pieces. It should have become clear that while this pattern uses more of the existing board, it is much more complicated and thus costly to cut.

In other problems, integrality does not occur naturally but is a result of the way we must formulate constraints. Some of these formulations use so-called *logical variables*, which are zero-one variables that are introduced to model logical implications that cannot be modeled in the usual variables. Recall that in a standard linear programming problem with, say, 100 constraints, it is obvious that all constraints must hold, otherwise there would be no reason to include them in the problem in the first place. There are, however, instances, that require a different treatment. For instance, if we have the choice of one of a number of different machines, each with its own capacity, then the actual capacity constraint is not given by all machines, but only by the one that is actually chosen. Such a requirement belongs to the class of so-called *either-or constraints*. Another class is that of *conditional constraints*. They can be spotted by their unique wording "*if* this, *then* that." Examples of these conditions will follow.

4.2.2 Diet Problems Revisited

First recall the standard diet problem in linear programming. To simplify matters, consider only two foodstuffs and a single nutrient. The quantities of the two foods are defined as x_1 and x_2, respectively, and at least five units of the nutrient are required in the diet. We assume that the problem has been formulated as follows.

$$\begin{aligned} \text{Min } z &= 3x_1 + 4x_2 \\ \text{s.t.} \quad & x_1 + 2x_2 \geq 5 \\ & x_1, \quad x_2 \geq 0. \end{aligned}$$

Suppose now that the additional requirement is that if food 1 is included in the diet (in any quantity), then food 2 should not be. (One of the reason for this may be incompatibilities due to taste such as ice cream and mustard, or unfortunate side effects of incompatible foods, such as water and green apples, or, worse, yoghurt and yeast). This is a conditional constraint of the type "if food 1 is included, then food 2 should not be." We first must define logical zero-one variables, one for each foodstuff. These new variables y_1 (and y_2) are defined as being one, if food 1 (food 2) is included in the diet, and zero otherwise. We will be needing these variables *in addition to* the variables x_1 and x_2 that denote the quantities of the two foods that are included in the diet.

In order to model this situation, it is beneficial to use a small table that includes all relevant solutions. Table 4.2 includes not only all combinations of the two foods, but also a column that indicates whether or not the solution is acceptable. For instance, the first row includes neither of the two foods, and while it may leave us hungry, it does not violate the condition. Here, we find that only the solution that has y_1 and y_2 both equal to one (the case in which both foods are in the diet) is prohibited.

Table 4.2: Decision table for the diet problem

y_1	y_2	OK?
0	0	✓
0	1	✓
1	0	✓
1	1	No

Eliminating this solution from consideration (this is where operations research becomes a bit of an art, rather than a science) is to write the constraint

$$y_1 + y_2 \leq 1.$$

This surely excludes only the case of both foods in the diet, and adding this constraint to our formulation should do it. However, it does not. The reason is that by adding this constraint to the formulation, our new problem now has two sets of constraints; one that includes only the continuous variables x_1 and x_2, and another, completely separate part that includes only the variables y_1 and y_2. This would allow the two types of variables to change their values independent of each other, for instance allowing solutions that have $y_1 = 0$ and $x_1 = 7$. This does not make sense, as $y_1 = 0$ states that "food 1 is not included in the diet," while $x_1 = 7$ says that "there are 7 units of food 1 in the diet."

The remedy is to include additional linking constraints, i.e., constraints that include the continuous variables x_1 and x_2 as well as the logical variables y_1 and y_2. In this problem, the linking constraints are

$x_1 \leq My_1$ and
$x_2 \leq My_2$,

where M is a "sufficiently large" constant. In order to understand the workings of these linking constraints, consider the first of these constraints and use the two possible solutions $y_1 = 1$ and $y_1 = 0$. If $y_1 = 1$, then the constraint reads $x_1 \leq M$, which, with M being very large, is a redundant constraint that does not affect the solution. If, on the other hand, $y_1 = 0$, then the constraint reads $x_1 \leq 0$, which, in conjunction with the nonnegativity constraint $x_1 \geq 0$, forces x_1 to be equal to zero. This is exactly the desired effect, as if food 1 is not included in the diet (i.e., $y_1 = 0$), then its quantity in the diet (x_1) must be zero as well. And this is what these linking constraints guarantee.

Consider now a few extensions of the model. If, for instance, it would not be acceptable to have a diet without any food (i.e., we were to consider the solution $y_1 = y_2 = 0$ unacceptable), then the constraint $y_1 + y_2 = 1$ would guarantee that while both foods cannot be together in the diet, at least one of them has to be.

Consider now the diet problem with the conditional constraint "if food 1 is included in the diet, then food 2 must be included in the diet as well." the decision table for this condition is shown in Table 4.3.

Table 4.3: Decision table for the modified diet problem

y_1	y_2	OK?
0	0	✓
0	1	✓
1	0	No
1	1	✓

For this scenario, the constraint

$y_1 \leq y_2$

must be included (together with the linking constraints). The only solution that this constraints excludes is $\mathbf{y} = (0, 1)$, which is the desired effect.

4.2.3 Land Use

The next example deals with land use. Suppose that a land owner owns a parcel of land that he has to decide what to do with. He has narrowed down his decisions to two: either sell stumpage, i.e., harvest the land, or build an animal sanctuary, but not both. Here, we will formulate only this aspect of the model and ignore all other considerations. We will need a decision variable for each possible decision, so that we define $y_1 = 1$, if we decide to harvest the parcel, and 0 otherwise, and $y_2 = 1$, if we decide to build an animal sanctuary, and 0 otherwise. The decision table for this problem is then shown in Table 4.4.

Table 4.4: Decision table for the simple land use problem

y_1	y_2	OK?
0	0	✓
0	1	✓
1	0	✓
1	1	No

Once formalized as done here, we see that the situation is the same as in the diet problem by not allowing both options at the same time, which is modeled as $y_1 + y_2 \leq 1$. Since this problem does not require quantitative variables x_1 and x_2 as was the case in the diet problem, we do not need linking variables, as there is nothing to link.

Things may get much more complicated when more options exist. Suppose now that for the parcel in question, three choices have been identified: Harvest (decision variable y_1), build a sanctuary (decision variable y_2), or allow the building of a municipal well (decision variable y_3). As in the previous land use example, it is not possible to harvest and have a sanctuary at the same time in the parcel in questions. Furthermore, the parcel cannot be harvested if there is a municipal well on the parcel, while we could very well have a well and a sanctuary on the same parcel. The decision table for this extended problem is shown in Table 4.5.

Table 4.5: Decision table for the extended land use problem

y_1	y_2	y_3	OK?
0	0	0	✓
0	0	1	✓
0	1	0	✓
1	0	0	✓
0	1	1	✓
1	0	1	No
1	1	0	No
1	1	1	No

The modeling of this situation is considerably more complicated than that in the previous examples. As a matter of fact, we need two constraints to ensure that the bottom three solutions in Table 4.5 are excluded from consideration. We first formulate the constraint $y_1 + y_2 \leq 1$, which eliminates the solutions in the last two rows of the decision table, while, among the remaining six solutions, the constraint $y_1 + y_3 \leq 1$ eliminates the solution (1, 0, 1). This leaves the first five solutions as options, which is what the decision maker had in mind.

4.2.4 Modeling Fixed Charges

The purpose of this subsection is to introduce the reader to decision models that provide options regarding the choice of machines, so as to only apply the machine capacity constraints of those machines that are purchased or leased. In particular, consider a publishing company that intends to produce its annual lineup of operations research texts. This year, they have the books by Gabby and Blabby (*GB*), Huff, Fluff, and Stuff (*HFS*), and the "Real OR" (*ROR*) texts. As usual nowadays, authors are required to do everything except for the for the printing, binding (and the subsequent marketing). A number of different machines can be leased for the printing and binding. The three printing machines under consideration are P_1, P_2, and P_3, while the two binding machines are B_4 and B_5. The processing times for the different books on the respective machines in minutes per book are shown in Table 4.6.

Table 4.6: Processing times for printing and binding machines

	P_1	P_2	P_3
GB	3	6	4
HFS	2	3	3
ROR	4	5	5

	B_4	B_5
GB	10	10
HFS	12	11
ROR	15	14

The capacities of the three printing machines are 120 100, and 110 hours (7,200, 6,000, and 6,600 minutes) in the planning period. Similarly, the capacities of the binding machines are 333⅓ and 300 hours, respectively (or 20,000 and 18,000 minutes). The costs to lease the machines are independent of the number of books made with them. They are \$10,000, \$8,000, \$9,000, \$20,000, and \$23,000, respectively. The profit contributions of the three books (other than the leasing costs) have been identified as \$40, \$60, and \$70. It has also been determined that the publishing house should produce at least 500 copies of the landmark *ROR* book in order to maintain a good academic image.

We can formulate the problem by first defining variables x_1, x_2 and x_3 that indicate the number of books of the three types that are manufactured and sold. (As usual in single-period models, we assume that all units that are manufactured can also be sold). In addition, we also need zero-one variables that show whether or not a machine is leased. In particular, we define binary variables y_1, y_2, …, y_5 that assume a value of one, if a machine is leased, and 0 otherwise. The objective function then consists of two parts: the sum of the profit contributions of the individual books (which are $40x_1$, $60x_2$, and $70x_3$), and the sum of the leasing costs, which are $10{,}000y_1$, $8{,}000y_2$, $9{,}000y_3$, $20{,}000y_4$, and $23{,}000y_5$, respectively.

First the easy constraints: making at least 500 copies of *ROR* is modeled as $x_3 \geq 500$, and the fact that we need at least one printing and one binding machine can be written as $y_1 + y_2 + y_3 \geq 1$ and $y_4 + y_5 \geq 1$, respectively.

Consider now the capacity constraints of the machines. As usual, they are written as (machine usage) ≤ (machine capacity). For example, for the first printing machine, the capacity constraint is $3x_1 + 2x_2 + 4x_3 \leq 7{,}200$. The problem here is that this constraint must hold only if the machine is actually leased. If it is not leased, the constraint can be ignored. The way to model this is the same technique that was applied in the diet problem to write the linking constraints. If the first machine is not leased, $y_1 = 0$, and the capacity constraint is made redundant, by having a sufficiently large right-hand side value. (Again, we will use the very large value M as introduced in linear programming). However, if the first machine is leased, we have $y_1 = 1$, and the right-hand side of the capacity constraint should be 7,200, the actual capacity of the first machine. We can formulate this by writing the right-hand side as $7{,}200 + M(1 - y_1)$.

To demonstrate the validity of this formulation, let $y_1 = 0$. In this case the right-hand side value is $7{,}200 + M(1 - y_1) = 7{,}200 + M$, so that the constraint has a very large right-hand side, making it redundant. On the other hand, if we do lease the first machine and $y_1 = 1$, the right-hand side value equals $7{,}200 + M(1 - y_1) = 7{,}200$, which is the actual capacity of the first machine. The formulation can then be written as follows.

P: Max $z = 40x_1 + 60x_2 + 70x_3 - 10{,}000y_1 - 8{,}000y_2 - 9{,}000y_3 - 20{,}000y_4 - 23{,}000y_5$
s.t. $3x_1 + 2x_2 + 4x_3 \leq 7{,}200 + M(1-y_1)$
$6x_1 + 3x_2 + 5x_3 \leq 6{,}000 + M(1-y_2)$
$4x_1 + 3x_2 + 5x_3 \leq 6{,}600 + M(1-y_3)$
$10x_1 + 12x_2 + 15x_3 \leq 20{,}000 + M(1-y_4)$
$10x_1 + 11x_2 + 14x_3 \leq 18{,}000 + M(1-y_5)$
$x_3 \geq 500$
$y_1 + y_2 + y_3 \geq 1$
$y_4 + y_5 \geq 1$
$x_1, x_2, x_3 \geq 0$ and integer
$y_1, y_2, y_3, y_4, y_5 = 0$ or 1.

Using a large value for the constant M (here, we use $M = 1{,}000{,}000$), multiplying the brackets and sorting the variables, we obtain constraints such as $3x_1 + 2x_2 + 4x_3 + 1{,}000{,}000y_1 \leq 1{,}007{,}200$ for the first constraint, and similar for the other four capacity constraints. The solution of this all-integer programming problem is $\bar{y}_2 = \bar{y}_4 = 1$ and $\bar{y}_1 = \bar{y}_3 = \bar{y}_5 = 0$ (i.e., we lease the second printing and the second binding machine), and make $\bar{x}_1 = 0$ *GB* books, $\bar{x}_2 = 1{,}039$ HFS books, and $\bar{x}_3 = 502$ ROR books. The profit associated with this plan is \$69,480. Note that the slack capacities on the printout will indicate huge (and meaningless) values for the machines that are not leased. This is due to the fact that their right-hand side values have the artificial value of $M = 1{,}000{,}000$, from which some nonexistant usage is subtracted.

4.2.5 Workload Balancing

The problem presented in this subsection deals with the allocation of tasks to employees, so as to ensure that none of the employees is overworked, while others are partially idle. We assume that tasks cannot be split, meaning that once an employee starts a job, he will have to finish it. Due to their different backgrounds and training, a job will take different amounts of time if different employees perform it. There are three workers W_1, W_2, and W_3, who will have to perform tasks T_1, …, T_5. Table 4.7 shows the processing times (in hours) for all worker – task combinations.

Table 4.7: Processing times for worker-task combinations

	T_1	T_2	T_3	T_4	T_5
W_1	5	1	9	4	9
W_2	4	3	8	3	8
W_3	7	5	6	4	7

In order to formulate the problem, we need to introduce zero-one variables, which are defined as $y_{ij} = 1$, if employee W_i is assigned to task T_j, and zero otherwise. The only constraints of the model ensure that each task is assigned to exactly one employee. Formally, we can write

$$y_{1j} + y_{2j} + y_{3j} = 1 \text{ for all } j = 1, ..., 5.$$

The more contentious issue concerns the objective function. First, we note that the actual working time of the employees can be written as

$$w_1 = 5y_{11} + 1y_{12} + 9y_{13} + 4y_{14} + 9y_{15},$$
$$w_2 = 4y_{21} + 3y_{22} + 8y_{23} + 3y_{24} + 8y_{25}, \text{ and}$$
$$w_3 = 7y_{31} + 5y_{32} + 6y_{33} + 4y_{34} + 7y_{35},$$

where the new variables w_1, w_2, and w_3 denote that time that employees W_1, W_2, and W_3 are busy working on the tasks. One possibility to ensure fairness in the solution is to attempt to make the longest working time as short as possible. In other words, the employee who works longest should have the shortest working hours possible. Formally for our problem, we can write this objective as

$$\text{Min } z = \max \{w_1, w_2, w_3\}.$$

Clearly, this is not part of a linear programming problem the way we have defined it. However, *minimax objective functions* of this type can easily be reformulated, resulting in standard linear (or integer) programming problems. This is done by

introducing a single new variable, say z, that measures the highest workload of any employee, i.e., the maximum of the right-hand side of the above objective. We will then minimize this highest workload, but then we must ensure that none of the actual workloads is higher, which is done by introducing the three constraints $z \geq w_1$, $z \geq w_2$, and $z \geq w_3$. Replacing w_1, w_2, and w_3 by the functions of y_{ij}, we can then write the model as

$$\begin{aligned}
&\text{Min } z \\
&\text{s.t. } z \geq 5y_{11} + 1y_{12} + 9y_{13} + 4y_{14} + 9y_{15} \\
&z \geq 4y_{21} + 3y_{22} + 8y_{23} + 3y_{24} + 8y_{25} \\
&z \geq 7y_{31} + 5y_{32} + 6y_{33} + 4y_{34} + 7y_{35} \\
&y_{11} + y_{21} + y_{31} = 1 \\
&y_{12} + y_{22} + y_{32} = 1 \\
&y_{13} + y_{23} + y_{33} = 1 \\
&y_{14} + y_{24} + y_{34} = 1 \\
&y_{15} + y_{25} + y_{35} = 1 \\
&y_{ij} = 0 \text{ or } 1 \text{ for } i = 1, 2, 3; j = 1, \ldots, 5.
\end{aligned}$$

The solution will have worker W_1 work on tasks T_2 and T_5, worker W_2 works on tasks T_1 and T_4, while worker W_3 is assigned task T_3. The resulting workloads for the three employees are then 10, 7, and 6 hours, respectively (with the variable $z = 10$ denoting the highest of the workloads).

Other objective functions to ensure fairness have been used in the literature. For instance, we could define a variable $w = w_1 + w_2 + w_3$ as the total workload by all of the employees, so that ideally, each employee would work $\frac{1}{3}w$. We could not minimize the sum of deviations from this workload. Since positive and negative deviations will cancel out, we have to either minimize the sum of absolute values of the deviations or square the deviations (as done to derive the variance in statistics, which is the sum of squared deviations from the mean). This objective would then be

$$\text{Min } z = ((1/3)w - w_1)^2 + ((1/3)w - w_2)^2 + ((1/3)w - w_3)^2.$$

Unfortunately, this objective is nonlinear and the resulting problem will then be nonlinear and integer, a very difficult combination. In general, fairness (or "equity") objectives should normally be combined with efficiency objectives. If not, a solution such as workloads of 12, 12, and 12 for three employees would be preferred to workloads of 4, 9, and 8 hours, because the former solution is "more equal," even though each employee would gain by moving from the former to the latter solution.

4.3 Solution Methods for Integer Programming Problems

This section will examine some of the techniques that can and are used to solve integer programming problems. The first subsection will briefly describe the ideas behind cutting plane methods. The second subsection thoroughly discusses the basic features of branch-and-bound techniques, and the third subsection illustrates some heuristic methods and how they may apply to some integer programming problems.

4.3.1 Cutting Plane Methods

As mentioned in the introduction to this chapter, the first exact techniques for the solution of integer programming problems were so-called *cutting plane techniques*. Their general idea may be explained as follows. Recall that some problems always have optimal integer solutions, such as the assignment problem and the transportation problem (the latter only if all supplies and demands are integer-valued). Given Dantzig's corner point theorem (see Section 2.3), this means that the feasible set of those two problems (and other like them) has corner points, all of which are integer-valued. Clearly, very few formulations have this property. If it were now feasible to find a restricted feasible set that includes all integer solutions of the original feasible set but has all of its extreme points integer-valued (something usually called a *convex hull*), then we could simply solve the linear programming relaxation of the problem with this restricted feasible set and then automatically obtain an integer solution. Unfortunately, obtaining the convex hull of a feasible set is very difficult and the suggested approach is not computationally viable.

However, it is not necessary to have the entire convex hull at our disposal. And this is where cutting plane methods come in. The idea of cutting plane methods is to locally approximate the convex hull. This is done as follows. We first solve the linear programming relaxation of the given problem. If the optimal solution is integer, we are done. Suppose now that it is not. We then formulate a cutting plane, i.e. an additional constraint that does two things: (1) it must cut off (i.e., make infeasible) the present optimal solution, while (2) it cannot cut off any feasible integer point.

As a numerical illustration, consider the following all-integer programming problem:

$$
\begin{aligned}
\text{P: Max } & z = y_1 + y_2 \\
\text{s.t. } & 3y_1 + 2y_2 \le 6 \\
& y_1 + 3y_2 \le 3 \\
& y_1, \quad y_2 \ge 0 \text{ and integer.}
\end{aligned}
$$

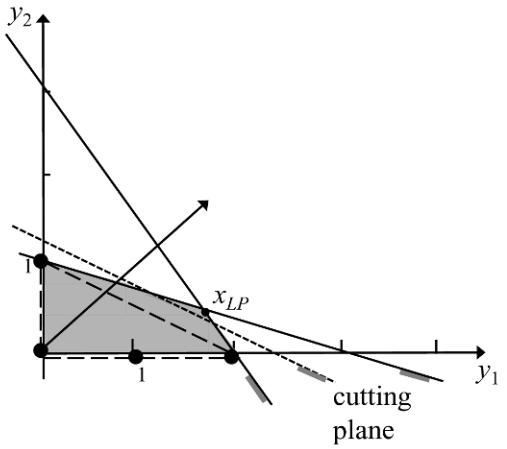


Figure 4.6

The shaded area in Figure 4.6 shows the feasible set of the linear programming relaxation, and the point shown as $\bar{\mathbf{y}}_{LP} = (12/7, 3/7)$ is the optimal solution of the linear programming relaxation. The triangle shown by the broken lines that connect the points (0, 0), (2, 0), and (0, 1) is the convex hull of the feasible set. The dotted line shows the cutting plane $5y_1 + 10y_2 \leq 12$. Plugging in the coordinates of $\bar{\mathbf{y}}_{LP}$, we see that this point violates the condition of the cutting plane, as $90/7 = 12\frac{6}{7} \not\leq 12$. This is also apparent in the graph. On the other hand, all four feasible integer points satisfy the condition, so that the condition is indeed a cutting plane.

While cutting planes appear to be a very good idea, their computational performance has been disappointing. In particular, the slices that are cut off tend to become tiny, requiring a very large number of additional constraints to be added in the process, which adds to the size of the problem and the degree of difficulty.

4.3.2 Branch-and-Bound Methods

In contrast to cutting plane methods, the concept of branch and bound in its many variations is nowadays accepted as the universal solver for mixed- and all-integer linear programming problems. The basic idea is simple. We first solve again the linear programming relaxation. Again, if its coordinates of the solution point satisfy the required integrality conditions, the process terminates. Suppose now that this is not the case. We then select any variable that is required to be integer but, at the present solution, is not. As an example, let the variable $y_7 = 3.4$ at the present solution. Given that y_7 is required to be integer, the present solution does not satisfy all integrality constraints. We will then subdivide the given problem (often called the "parent") into two new problems (the "children"), a process usually referred to as "divide and conquer" (or, by some critics, "double your trouble").

Each of the two children contains exactly the same constraints as its parent plus one single additional constraint. One child has the additional constraint $y_7 \leq 3$, while the other child has the additional constraint $y_7 \geq 4$. That way, we completely eliminate all solutions for which the variable $3 < y_7 < 4$. The reason that we can actually cut out such a "corridor" without deleting important parts of the feasible set is that none of the solutions that we eliminate from consideration are feasible, in that none of them contain any point that has an integer value for the variable y_7.

This way, we will build up what is known as a solution tree. Each "node" of the tree, shown as a small box, represents a problem formulation and a solution. Once we decide to work on a node or solution, we "branch" from it, meaning we generate its children by adding a constraint as explained above. Once we have branched from a node, it is no longer *active*. Also, nodes that represent integer solutions and those that represent problem formulations that do not have feasible solutions are also not active. This is best explained in terms of a small numerical example.

Consider the following all-integer optimization problem:

$$\begin{aligned} \text{Max } z = {} & 5y_1 + 9y_2 & & \\ \text{s.t.} \quad & 5y_1 + 11y_2 \leq 94 & & \text{Constraint } I \\ & 10y_1 + 6y_2 \leq 87 & & \text{Constraint } II \\ & y_1, \quad y_2 \geq 0 \text{ and integer.} & & \end{aligned}$$

The graphical representation of the problem is shown in Figure 4.7, where the feasible set of the linear programming relaxation is the shaded area. The optimal solution of the linear programming relaxation is depicted as the point $\bar{\mathbf{y}}_{LP}$. The intermediate results of our computations will be displayed and collected in the solution tree shown in Figure 4.8. At present, only the top node, known as the *root of the tree*, is known. At this point, this is also the only active node.

The first step is to examine the solution. In this example, both variables are supposed to be integer and neither of them is. This means that we can start working on either variable. Here, we choose to start working on y_2. (Alternatively, we could have started working on y_1. This will result in a completely different solution tree—which one is smaller and easier cannot be said in advance—which is shown in Figure 4.9). At present, $y_2 = 6.3125$, so that we will eliminate the corridor $6 < y_2 < 7$, so that our first branching, shown in the solution tree in Figure 4.8 branches from the root tree to the two children, the one on the left with the additional constraint $y_2 \leq 6$, and the one on the right with the additional constraint $y_2 \geq 7$. In Figure 4.7, the feasible set of the right child is now the triangle with vertices at (0, 7), (0, 8.5455), and $\bar{\mathbf{y}}^2 = (3.4, 7)$. Using the objective function, the optimal solution of this subproblem is $\bar{\mathbf{y}}^2$. On the other hand, the feasible set of

the left child is the trapezoid with corner points (0, 0), (0, 6), $\bar{\mathbf{y}}^1 = (5.1, 6)$, and (8.7, 0). The optimal solution of this subproblem is $\bar{\mathbf{y}}^1$.

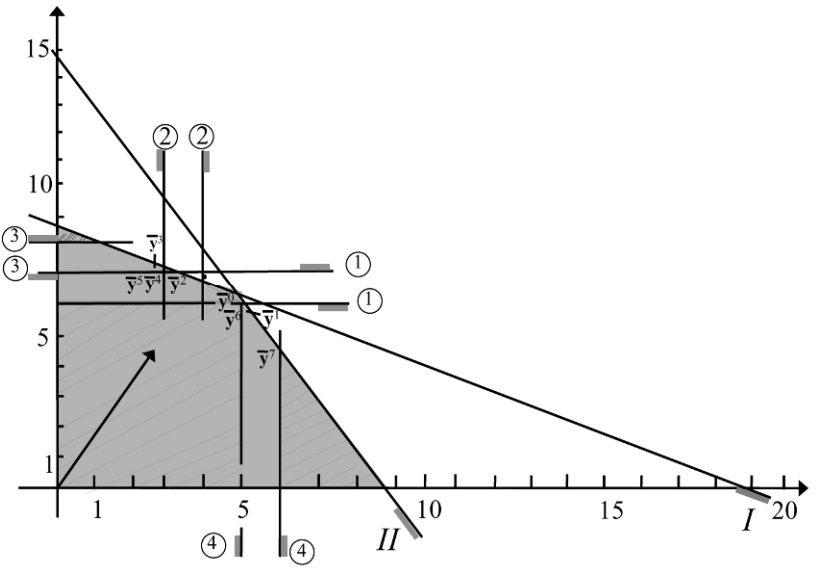


Figure 4.7

At this point we have two active nodes with solutions $\bar{\mathbf{y}}^1$ and $\bar{\mathbf{y}}^2$. Among all active nodes, we always choose the one with the best value of the objective function (i.e., highest for maximization problems and lowest for minimization problems). In this example, it is the node with the solution $\bar{\mathbf{y}}^2$. This is the node (now considered the parent) that we will examine and further work on, if necessary. The solution $\bar{\mathbf{y}}^2$ has values of 3.4 and 7 for the two variables, meaning that while the variable y_2 is integer as required, the variable y_1 is not and we will have to work on it. This means that from this solution, we will perform our second branching that adds the constraint $y_1 \leq 3$ to the left child, and $y_1 \geq 4$ to the right child. The additional constraints are shown in Figure 4.7 as small lines that will have to be considered in conjunction with the triangle (the feasible set of the parent) derived earlier. This results in a feasible set of the left child that is the trapezoid with corner points (0, 6), (0, 8.5455), $\bar{\mathbf{y}}^3 = (3, 7.1818)$, and (3, 6), the optimal solution of which is $\bar{\mathbf{y}}^3$. On the other hand, the right child has an empty feasible set, as the intersection of the triangle of its parent and the constraint $y_1 \geq 4$ is empty. This is now indicated in the solution tree in Figure 4.8. At this point we have two active nodes: the nodes labeled with solutions $\bar{\mathbf{y}}^1$ and $\bar{\mathbf{y}}^3$. All other

nodes developed so far have either already been branched from or have no feasible solution and thus need no longer to be considered.

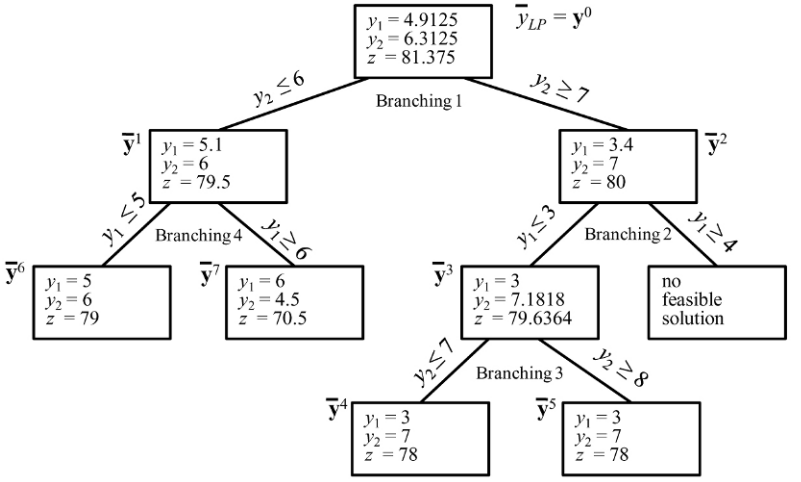


Figure 4.8

The node with the best (here: highest) value of the objective functions value is $\bar{\mathbf{y}}^3$, so that our work continues here. Now the variable $y_2 = 7.1818$ is again noninteger and we will branch on it. Considering this node as the present parent, we branch to the children by using the additional constraints $y_2 \le 7$ and $y_2 \ge 8$, respectively. The feasible set of the left child is the line segment with end points (0, 7) and (3, 7). The optimal solution of this child is $\bar{\mathbf{y}}^4 = (3, 7)$ with a value of the objective function of 78. This is our first integer solution so far in the tree. The feasible set of the right child is the triangle with vertices (0, 8), (0, 8.5455), and $\bar{\mathbf{y}}^5$, with $\bar{\mathbf{y}}^5$ being its optimal solution, whose objective value is also 78.

At this point, the active nodes are $\bar{\mathbf{y}}^1$, $\bar{\mathbf{y}}^4$, and $\bar{\mathbf{y}}^5$. Inspection reveals that $\bar{\mathbf{y}}^1$ has the highest value of the objective function, so that work will continue here, regardless of the fact that we have already found an integer solution (that solution may not be the best). This node is now temporarily considered the parent. In this solution, $y_1 = 5.1$, which violates the required integrality condition.

Recall that the feasible solution of this parent is the shaded area below $y_2 \le 6$. Branching $y_1 \le 5$ and $y_1 \ge 6$, we obtain the two crosshatched feasible sets: the rectangle with vertices (0, 0), (0, 6), $\bar{\mathbf{y}}^6 = (5, 6)$, and (5, 0) for the left child, and the triangle with vertices (6, 0), $\mathbf{y}^7 = (6, 4.5)$, and (8.7, 0) for the right child. The respective optimal solutions are $\bar{\mathbf{y}}^6$ and $\bar{\mathbf{y}}^7$, whose values of the objective function are 79 and 70.5, respectively.

At this point, we have four active nodes with solutions $\bar{\mathbf{y}}^4$, $\bar{\mathbf{y}}^5$, $\bar{\mathbf{y}}^6$, and $\bar{\mathbf{y}}^7$. The node with the best value of the objective function is $\bar{\mathbf{y}}^6$, whose solution is $\bar{\mathbf{y}}^6 = (5, 6)$ and objective value 79. Work will continue on this node. Inspection reveals that all integrality conditions are now satisfied, so that the $\bar{\mathbf{y}}^6$ represents an optimal solution. Since no other active node has an equal objective value, this optimal solution is unique, so that the problem is solved.

We have seen that the branch-and-bound method solves integer programming problems by a sequence of linear programming problems. It is not at all uncommon that one integer programming problem requires the solution of hundreds of thousands of linear programming problems, all similar, but differing by a few constraints. Consider again the solution tree in Figure 4.8 for some additional considerations. For instance, the additional constraints to be considered at a node are the constraints at all branchings from the root of the tree to the node under consideration. For instance, the node with optimal solution $\bar{\mathbf{y}}^5$ represents a problem with the given objective function, all constraints of the linear programming relaxation (here: constraints *I* and *II*), as well as the additional constraints $y_2 \geq 7$, $y_1 \leq 3$, and $y_2 \geq 8$. An immediate consequence of this principle leads to the observation that as you move on some path down the tree, the values of the objective function of the problems either stay the same or get worse (i.e., go down for maximization problems and go up for minimization problems). This occurs as by adding constraints, the feasible set gets smaller, which cannot result in better objective values.

It is also apparent that even a very small problem with only two variables can have sizeable solution trees. When using a computer with linear programming software installed, it is possible to practice by using the computer to solve the linear programming problems at the nodes of the solution tree, while constructing the solution tree manually. In order to solve the integer programming problem in such "computer-assisted" fashion, it is necessary to learn how to edit the linear programming problems. For instance, return from the problem with the solution $\bar{\mathbf{y}}^5$ to $\bar{\mathbf{y}}^1$ requires moving up to the root of the tree (thus removing the constraints $y_2 \geq 8$, $y_1 \leq 3$, and $y_2 \geq 7$, and then moving down to the node labeled $\bar{\mathbf{y}}^1$ by adding the constraint $y_2 \leq 6$.

For practice, readers may either use the graphical approach or the computer-assisted approach to solve the same problem but start branching at the root node on the variable y_1 rather than y_2. This will result in the solution tree shown in Figure 4.9. Note that this tree requires six branchings and the optimal solution is found at the node labeled $\bar{\mathbf{y}}^9$.

It is also important to realize that the very same procedure can be used to solved mixed-integer linear programming problems. Whenever the best active node has been chosen, we have to compare what is required to be integer and what is. As an illustration, use the same example as above, except that $y_1 \geq 0$, while $y_2 \geq 0$ and integer. Again, we first solve the linear programming relaxation and find the root node of the solution tree. Now as $y_1 = 4.9125$, it satisfies the nonnegativity constraint, which is all that is required of this variable. On the other hand, $y_2 = 6.3125$, which satisfies the nonnegativity constraint, but not the integrality constraint, so we must branch on y_2 (there is no choice as in the all-integer problem). The branching is identical to the first branching shown in Figure 4.8. At this point, $\bar{\mathbf{y}}^2$ is the better solution with $y_1 = 3.4$, $y_2 = 7$ with an objective value of 80. At this point, the procedure terminates, as y_1 is nonnegative as required, and y_2 is also nonnegative and integer, as required; hence $\bar{\mathbf{y}}^2$ is the optimal solution for this problem.

This is also the time to demonstrate what the term "bound" in branch-and-bound actually means. The idea is that during the procedure, the upper and lower bounds on the optimal objective value of the all- or mixed-integer programming problem are constantly tightened. Consider the process shown in Figure 4.9. Initially, as shown in the first section of this chapter, we only know that $\bar{z}_{IP} \leq \bar{z}_{LP}$, i.e., the optimal objective value of the integer problem is no better than 81.375. For maximization problems, the upper bound is always the objective value of the best known active node, while the lower bound is the objective value of the best known integer solution. Hence, initially $-\infty \leq \bar{z}_{IP} \leq 81.375$. After the first branching, the upper bound has decreased to 80.5455. After the second branching, the upper bound is reduced to 80.5, and the lower bound is now 74, as the integer solution $\bar{\mathbf{y}}^3$ is now known. Branching 3 further reduces the upper bound to 80, branching 4 reduces the upper bound to 79.6364, branching 5 reduces the upper bound to 79.5, while the lower bound is now increased to 78, as the integer solution $\bar{\mathbf{y}}^7$ has become known. Finally, after branching 6 the upper and lower bounds coincide at 79, which terminates the process.

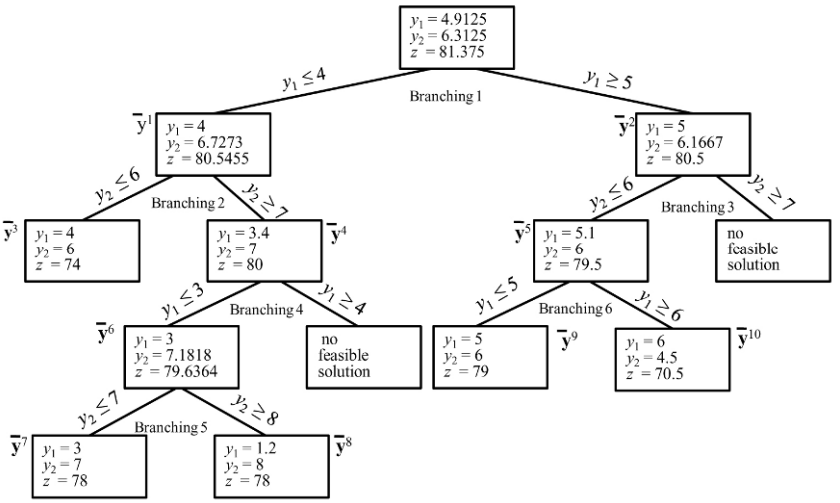


Figure 4.9

We will now demonstrate how to deal with cases in which the mixed- or all-integer programming problem has no feasible solution. As an illustration, consider the all-integer programming problem

$$\begin{aligned} \text{P: Max } z &= y_1 + 4y_2 \\ \text{s.t.} \quad & 28y_1 + 7y_2 \le 49 \\ & 30y_1 - 6y_2 \ge 36 \\ & y_1, \quad y_2 \ge 0 \text{ and integer.} \end{aligned}$$

The solution tree for this problem, given that branching commences with y_2, is shown in Figure 4.10. Branching 1 results in one child of the root having no feasible solution, while the other has a noninteger solution. Branching on this sole active node, we obtain two children, both of whom have no feasible solutions. At this point, there are no further active nodes and no solution has been found. The tree that is obtained by start branching on y_1 is even smaller: the two children of the root of the tree both have no feasible solution.

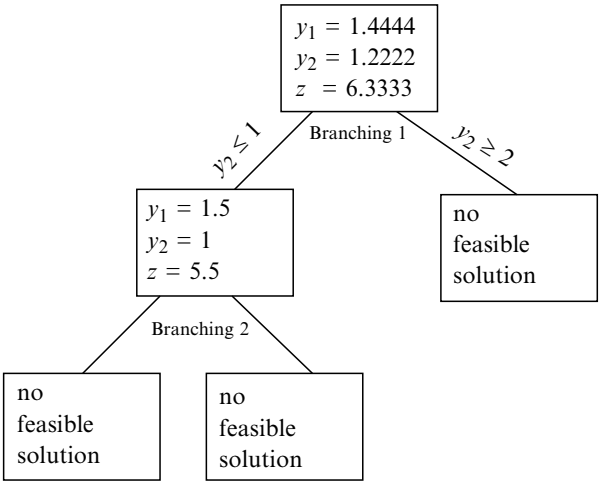


Figure 4.10

4.3.3 Heuristic Methods

It will have become clear in the above discussion that for very large problems, exact methods may not be able to solve a given integer linear programming problem within a reasonable time frame. Surely, whenever an exact method such as the branch-and-bound technique described in the previous section has found an integer solution, one could terminate computations, even though the integer solution may not be optimal. As a matter of fact, while such a procedure will result in a feasible solution, it may be far from optimal. Instead, users often use heuristic techniques to find (hopefully reasonably good) solutions quickly. This section will describe such a technique.

In order to illustrate the technique, consider the following knapsack problem:

$$\begin{aligned} \text{P: Max } z &= 12y_1 + 20y_2 + 31y_3 + 17y_4 + 24y_5 + 29y_6 \\ \text{s.t.} \quad & 2y_1 + 4y_2 + 6y_3 + 3y_4 + 5y_5 + 5y_6 \leq 19 \\ & y_1, \quad y_2, \quad y_3, \quad y_4, \quad y_5, \ldots \; y_6 = 0 \text{ or } 1. \end{aligned}$$

In order to employ the *Greedy Method* (sometimes colloquially referred to as "bang for the buck" method), we first need to rank the variables in nonincreasing order of their "value" to us. Rather than simply using the coefficients in the objective function to rank the variables, we compute the "value per weight" ratios for each product by dividing the contribution to the objective function by the coefficient in the constraint. We then rank the variables in nonincreasing order of these ratios as shown in Table 4.8.

Table 4.8: "Value per weight" of the individual items

Variable	y_1	y_2	y_3	y_4	y_5	y_6
Value per weight	12/2 = 6	20/4 = 5	31/6 = 5.1667	17/3 = 5.6667	24/5 = 4.8	29/5 = 5.8
Rank	1	5	4	3	6	2

Starting with all variables set to zero, the Greedy algorithm will now increase the values of variables one by one, starting with the highest rank, as long as resources are available.

In Step 1, we set $y_1 := 1$, which consumes 2 resource units and contributes 12 units to the objective,
in Step 2, we set $y_6 := 1$, which consumes 5 resource units and contributes 29 units to the objective,
in Step 3, we set $y_4 := 1$, which consumes 3 resource units and contributes 17 units to the objective,
in Step 4, we set $y_3 := 1$, which consumes 6 resource units and contributes 31 units to the objective,
in Step 5, we set $y_2 := 1$, which consumes 4 resource units. Stop and backtrack, as this latest assignment exceeds the availability of resources.

In summary, the solution at the termination of the Greedy algorithm is

$$\mathbf{y} = [y_1, y_2, y_3, y_4, y_5, y_6] = [1, 0, 1, 1, 0, 1]$$

which uses 16 resource units and has a value of the objective function of $z(\mathbf{y}) = 89$.

In the second phase, we use a simple improvement heuristic of the "interchange" or "swap" type. This heuristic method raises one arbitrarily chosen variable from a level of zero to one, while decreasing another also arbitrarily chosen variable from one to zero. Whenever such an exchange is feasible and increases the value of the objective function, it becomes our new starting point. The procedure is reapeated until no further improvements are possible. Note that given the present solution, we have an extra 3 resource units available. In the tables below, we list the variable that leaves the solution (i.e., the variable whose value is reduced from its present value of one to zero), the entering variable (i.e., the variable whose value is increased from its present value of zero to one), the resulting new solution, the marginal resource usage ΔR, and the change Δz of the objective function that results from the swap.

Table 4.9: First swap move

Leaving variable	Entering variable	New solution	ΔR	Δz
y_1	y_2	0, 1, 1, 1, 0, 1	–2 + 4 = +2	–12 + 20 = + 8

It is apparent that the first swap move in Table 4.9 results in an improvement, so that the solution $\mathbf{y}$ = [0, 1, 1, 1, 0, 1] becomes the new basis with resource consumption of 18 (so that we could use one additional resource unit) and objective value of $z(\mathbf{y}) = 97$. Starting with this solution, we perform again pairwise exchanges shown in Table 4.10.

Table 4.10a: Second swap move

Leaving variable	Entering variable	New solution	ΔR	Δz
y_2	y_1	1, 0, 1, 1, 0, 1	–4 + 2 = –2	–20 + 12 = –8
y_2	y_5	0, 0, 1, 1, 1, 1	– 4 + 5 = 1	–20 + 24 = +4

Again, it was possible to improve the solution. The new solution is $\mathbf{y}$ = [0, 0, 1, 1, 1, 1], which uses all 19 resource units that are available, and has a value of the objective function of $z(\mathbf{y})$ = 101. With this new benchmark solution, we start another series of potential improvements.

Table 4.10b: Third swap move

Leaving variable	Entering variable	New solution	ΔR	Δz
y_3	y_1	1, 0, 0, 1, 1, 1	– 6 + 2 = –4	–31 + 12 = –19
y_3	y_2	0, 1, 0, 1, 1, 1	–6 + 4 = –2	–31 + 20 = –11
y_4	y_1	1, 0, 1, 0, 1, 1	– 3 + 2 = – 1	–17 + 12 = –5
y_4	y_2	0, 1, 1, 0, 1, 1	–3 + 4 = +1: infeasible	
y_5	y_1	1, 0, 1, 1, 0, 1	–5 + 2 = –3	–24 + 12 = –12
Y_5	y_2	0, 1, 1, 1, 0, 1	–5 + 4 = –1	–24 + 20 = –
y_6	y_1	1, 0, 1, 1, 1, 0	–5 + 2 = –3	–29 + 12 = –17
y_6	y_2	0, 1, 1, 1, 1, 0	–5 + 4 = –1	–29 + 20 = –9

At this point, all feasible pairwise exchanges result in decreases of the value of the objective function, so that the procedure terminates. Note that the fact that all resource units happen to be used is a coincidence.

It is also worth pointing out that the Greedy procedure alone may result in very poor solutions. As an example, consider the problem

$$\begin{aligned} \text{P: Max } z &= 10y_1 + 8y_2 + 7y_3 \\ \text{s.t.} \quad & 54y_1 + 48y_2 + 47y_3 \le 100 \\ & y_1, \quad y_2, \quad y_3 = 0 \text{ or } 1. \end{aligned}$$

The ranking of the variables is y_1, y_2, and y_3 in that order. Setting the highest-ranked variable y_1 to 1 consumes 54 resource units and achieves an objective value of z = 10. No other variables can be set to one, as only 46 resource units remain. However, the solution $\mathbf{y}$ = [0, 1, 1] has an objective value of z = 15, much

superior to the solution obtained by the Greedy algorithm (without subsequent Swap procedure). To see that Greedy alone without any improvement follow-up may reach an objective value of only half the optimum, consider the problem

$$\begin{aligned} \text{P: Max } z &= 1.01y_1 + 2y_2 \\ \text{s.t.} \quad & 1.001y_1 + 2y_2 \leq 2 \\ & y_1, \quad y_2 = 0 \text{ or } 1. \end{aligned}$$

The Greedy algorithm will find the solution $\mathbf{y} = [1, 0]$ with the objective value of $z = 1.01$, while the optimal solution is $\mathbf{y} = [0, 1]$ with the value of the objecive function $z = 2$.

Exercises

Problem 1 (a one-dimensional cutting stock problem): A home building store faces the following problem. Their customers demand half inch plywood in the sizes 4 ft ×3 ft, 4 ft × 5 ft, and 4 ft × 6 ft. Customer demand for these three sizes is estimated to be at least 20, 50, and 40 and customers are prepared to pay \$7, \$9, and \$10 for each of these sheets, respectively. The store must generate these sizes by cutting up standard 4 ft × 8 ft sheets, each of which costs them \$6. These sheets are in unlimited supply. Furthermore, each cut costs the store \$1.50. Formulate a model that indicates to the decision maker how to cut up the 4 ft × 8 ft sheets so as to maximize their profit. Clearly show the cutting patterns and define your variables properly.

Solution: Since the widths are all 4 ft, we only have to consider a single dimension. The cutting plan includes patterns with two 3 ft × 4 ft boards (and the resulting 2 ft × 4 ft piece of waste), one 3 ft × 4 ft board and a 5 ft × 4 ft board without any waste, and a single 6 ft × 4 ft board with the resulting 2 ft × 4 ft piece of waste. The number of cuts performed according to these three patterns are denoted by y_1, y_2, and y_3, respectively. The problem can then be formulated as

$$\begin{aligned} \text{P: Max } z &= [7(2y_1 + y_2) + 9(y_2) + 10\,(y_3)] - 6[y_1 + y_2 + y_3] - 1.5[2y_1 + y_2 + y_3] \\ &= 5y_1 + 8.5y_2 + 2.5y_3 \end{aligned}$$

$$\begin{aligned} \text{s.t. } 2y_1 + y_2 &\geq 20 \qquad \text{(generate at least twenty 4 ft} \times \text{3 ft sheets)} \\ y_2 &\geq 50 \qquad \text{(generate at least fifty 4 ft} \times \text{5 ft sheets)} \\ y_3 &\geq 40 \qquad \text{(generate at least forty 4 ft} \times \text{6 ft sheets)} \\ y_1, \quad y_2, y_3 &\geq 0. \end{aligned}$$

Problem 2 (a 2-dimensional cutting stock problem): A planner has 30 sheets of plywood of size 10 ft × 10 ft. They presently need 20 sheets in the shape of disks of diameter 5 ft as well as 15 sheets of plywood in the shape of 4 ft × 6 ft rectangles. The cutting patterns that are considered by the decision maker are shown in Figure 4.11.

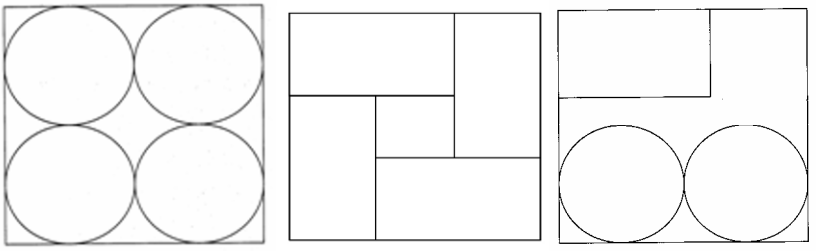


Figure 4.11

It costs \$2.00 to cut a disk and \$1.50 to cut a rectangle. This price includes all required cuts and is independent of the pattern the shape is cut from. Alternatively, we could purchase a disk at \$4.50 and a rectangle for \$3.00 each.

Formulate a linear programming problem that

- minimizes the cost of obtaining the required shapes,
- does not use more 10 ft × 10 ft sheets than are available
- produces the required numbers of disks and rectangles, and
- ensures that the cutting results in no more than 30% of waste.

Define all variables clearly.

Solution: Define y_j, $j = 1, 2, 3$ as the number of times the j-th pattern is cut. In addition, define variables y_4 and y_5 as the number of disks and rectangles that are purchased.

$$
\begin{array}{lrrrrrl}
\text{Min } z = & 8y_1 + & 6y_2 + & 5.5y_3 + & 4.5y_4 + & 3y_5 & \\
\text{s.t.} & y_1 + & y_2 + & y_3 & & & \le 30 \\
& 4y_1 + & & 2y_3 + & y_4 & & \ge 20 \\
& & 4y_2 + & 1y_3 + & & y_5 & \ge 15 \\
& .2146y_1 + & .04y_2 + & .3673y_3 & & & \le .3(y_1 + y_2 + y_3) \\
& y_1, & y_2, & y_3, & y_4, & y_5 \ge 0. &
\end{array}
$$

Problem 3 (neighborhood constraints in forestry modeling): This model concerns planning in forestry. In particular, suppose that a land owner has a number of parcels, which, for the sake of simplicity, can be thought of as a regular grid. The idea is now to plan which of the parcels should be harvested. Once it has been decided to harvest a certain parcel, it will be clearcut. One restriction is to ensure that neighboring parcels should not be harvested, so as to ensure avoiding huge clearcut areas that foster erosion and do not sustain wildlife. Suppose that the parcels are arranged and numbered as shown in Figure 4.12.

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

Figure 4.12

Model the neighborhood constraints that ensure that neighboring parcels are not harvested.

Solution: Define binary variables y_j that assume a value of 1, if parcel j is harvested and 0 otherwise. The constraints we are interested in can then be expressed as conditional constraints of the type "if parcel *j* is harvested, then its neighbor *k* cannot be harvested," which will then have to be expressed for all neighbors of each parcel. As an example, consider the neighboring parcels 1 and 2. Table 4.11 shows the four cases.

Table 4.11: Decision table

y_1	y_2	OK?
0	0	✔
0	1	✔
1	0	✔
1	1	No

It is apparent that the only case that is prohibited is when both neighboring parcels are harvested at the same time. As a result, the constraint that prevents the undesirable case from happening, we formulate $y_1 + y_2 \leq 1$. This type of constraint has to be formulated for *all* pairs of neighbors, e.g., parcels 1 and 2, 1 and 5, 2 and 3, 2 and 6, and so forth. For the above problem, there are no less than 24 such neighborhood constraints. If parcels that only share a corner are also considered neighbors (such as parcels 1 and 6 or parcels 8 and 11), that will add another 18 constraints.

Problem 4 (a warehouse distribution problem): A decision maker has presently three warehouses with capacities of 30, 10, and 50 units, respectively. Operating the three warehouses incurs fixed operating costs of \$25, \$50, and \$45, respectively. The first warehouse could be expanded by up to 20 units for a cost of \$1 per unit capacity. It is also possible to close any of the existing warehouses, in which case no operating costs are incurred at that site. Furthermore, two additional sites have been identified, where new warehouses might be opened. The costs to open the two new warehouses are \$15 and \$25, and their respective capacities are 20 and 30 units.

As far as demand goes, there are presently three customers with demands of 20, 60, and 40 units. These demands have to be satisfied exactly. The unit transportation costs between the existing and potential warehouses and the customers are shown in Table 4.12, where the first three supply points refer to the existing warehouses, while the last two supply points symbolize the potential new warehouses. Formulate and solve an integer programming problem that minimizes the total costs.

Table 4.12: Unit transportation costs

	Customer 1	Customer 2	Customer 3
Warehouse 1	3	7	4
Warehouse 2	2	6	8
Warehouse 3	9	3	4
New warehouse 1	5	6	4
New warehouse 2	7	3	9

Solution: In order to formulate the problem, we need to define a number of variables. First of all, we define binary variables y_1, y_2, y_3, y_4, and y_5, which assume a value of one, if the warehouse is kept open (for the first three existing warehouses), or is newly opened in case of the last two warehouses. Furthermore, we define the continuous variable w, which denotes the number of capacity units, by which the first warehouse is expanded. In addition, we will define the usual continuous variables x_{ij} as the quantity that is shipped from warehouse i to customer j.

The objective function is then the sum of facility costs (operating costs for the existing warehouses and opening costs for the planned new warehouses), expansion costs for the first warehouse, and transportation costs. The facility costs are, of course, only incurred if the facilities are actually opened, so that we have $25y_1 + 50y_2 + 45y_3$ for the existing warehouses and $15y_4 + 25y_5$ for the new warehouses. The expansion costs for the first warehouse are $1w$. Finally, the transportation costs are $3x_{11} + 7x_{12} + 4x_{13} + 2x_{21} + \ldots + 9x_{53}$.

Consider now the constraints. First there are the capacity constraints of the warehouses. For instance, the second warehouse has a capacity of 10, provided we keep it open. The variable that determines this is y_2, so that the constraint states that the flow out of warehouse 2 cannot exceed the capacity of the warehouse, or $x_{21} + x_{22} + x_{23} \leq 10y_2$. The constraints for the other existing or planned warehouses, except for the first, are similar. The capacity of the first warehouse equals $30y_1$ plus the capacity of the expansion (if any), which is w. We also have to specify that the expansion will only be undertaken if the warehouse is kept open, as it is meaningless to decide to close warehouse 1 and then expand its capacity. This is written as $w \leq 20y_1$. The reason is that if the warehouse is kept open, we have $y_1 = 1$, so that the constraint states that we can expand its capacity of 20 units. On the other hand, if warehouse 1 is closed, $y_1 = 0$ and the constraint

states that $w \leq 0$, meaning that no expansion is possible. The demand constraints then require the inflow into the customer sites to be equal to the demand at the site. The complete formulation is then as follows:

$$\begin{aligned}
\text{P: Min } z = {} & 25y_1 + 50y_2 + 45y_3 + 15y_4 + 25y_5 + 1w \\
& + 3x_{11} + 7x_{12} + 4x_{13} + 2x_{21} + 6x_{22} + 8x_{23} + 9x_{31} + 3x_{32} + 4x_{33} \\
& + 5x_{41} + 6x_{42} + 4x_{43} + 7x_{51} + 3x_{52} + 9x_{53}
\end{aligned}$$

$$\begin{aligned}
\text{s.t. } & x_{11} + x_{12} + x_{13} \leq 30y_1 + w \\
& x_{21} + x_{22} + x_{23} \leq 10y_2 \\
& x_{31} + x_{32} + x_{33} \leq 50y_3 \\
& x_{41} + x_{42} + x_{43} \leq 20y_4 \\
& x_{51} + x_{52} + x_{53} \leq 30y_5
\end{aligned}$$

$$w \leq 20y_1$$

$$\begin{aligned}
& x_{11} + x_{21} + x_{31} + x_{41} + x_{51} = 20 \\
& x_{12} + x_{22} + x_{32} + x_{42} + x_{52} = 60 \\
& x_{13} + x_{23} + x_{33} + x_{43} + x_{53} = 40
\end{aligned}$$

$$\begin{aligned}
& x_{ij} \geq 0 \text{ for all } i, j;\ w \geq 0 \\
& y_1, \ldots, y_5 = 0 \text{ or } 1.
\end{aligned}$$

Solving the problem results in keeping warehouses 1 and 3 open, while closing warehouse 2. In addition, warehouse 5 will be opened, while warehouse 4 is not. It was also decided to expand the capacity of warehouse 1 by 10 units. The shipments are as follows: from warehouse 1, we send 20 units to the first and 20 units to the third customer, from warehouse 3 we send 30 units to customer 2 and 20 units to customer 3, and from the newly opened warehouse 5 we ship 30 units to customer 2. The total cost of the operations are \$505.

Problem 5 (choosing vehicles for a display): An impresario wants to put up an exhibit featuring some antique cars. The vehicles potentially available are a Bugatti, Cadillac, Cobra, Corvette, Pierce Arrow, and Studebaker. The impact of the individual vehicles has been estimated in terms of the number of people who would make a special trip to see a vehicle as 58, 37, 42, 40, 55, and 33. The budget of the organizer is \$15,000, and the costs to transport the automobiles to the venue (they are presently located at different sites) and the costs of their insurance (depending on the vehicles' estimated value) are \$6,000, \$4,000, \$3,800, \$4,200, \$5,500, and \$3,200. The obvious idea is to choose vehicles for the exhibit, so as to maximize the impact, while staying within the budget. In addition, there are some further requirements.

- Choose at least three vehicles for the exhibit.
- If a Corvette is included in the exhibit, then a Cobra must also be included.
- If a Bugatti is not included in the show, then a Cadillac must be included.

Solution: In order to formulate the problem, we first define binary variables y_1, y_2, y_3, y_4, y_5, and y_6 that assume a value of one, if the 1st, 2nd, …, 6th vehicle is included in the exhibit, and 0 otherwise. The formulation of the objective function and the budget constraint are straightforward. Consider now the additional requirements. The number of vehicles included in the exhibit is expressed as the sum of all variables, so that we can write $y_1 + y_2 + y_3 + y_4 + y_5 + y_6 \geq 3$.

The next step is the conditional constraint "if Corvette, then Cobra," or, more formally, "if $y_4 = 1$, then $y_3 = 1$ as well."

y_4	y_3	OK?
0	0	✓
0	1	✓
1	0	No
1	1	✓

The undesirable solution $y_4 = 1$ and $y_3 = 0$ can be avoided by including the constraint $y_4 \leq y_3$, which violates the undesirable solution, while it is valid in the other three solutions.

Consider now the last requirement. The conditional constraint is "if not Bugatti, then Cadillac," or, equivalently, "if $y_1 = 0$, then $y_2 = 1$." Again, consider the decision table below.

y_1	y_2	OK?
0	0	No
0	1	✓
1	0	✓
1	1	✓

The only solution that violates the condition is the one that has neither of the two vehicles in the exhibit. In other words, at least one of the two vehicles must be in the exhibit, so we can formulate $y_1 + y_2 \geq 1$.

The formulation of the entire problem is then

$$\text{Max } z = 58y_1 + 37y_2 + 42y_3 + 40y_4 + 55y_5 + 33y_6$$

$$\begin{aligned}
\text{s.t.} \quad & 6{,}000y_1 + 4{,}000y_2 + 3{,}800y_3 + 4{,}200y_4 + 5{,}500y_5 + 3{,}200y_6 \leq 15{,}000 \\
& y_1 + y_2 + y_3 + y_4 + y_5 + y_6 \geq 3 \\
& y_4 \leq y_3 \\
& y_1 + y_2 \geq 1 \\
& y_1, y_2, y_3, y_4, y_5 \ldots, y_6 = 0 \text{ or } 1.
\end{aligned}$$

Table 4.13 displays solutions and objective values for a large variety of budgets. It is apparent that there is no feasible solution for any budget strictly less than 11, as we have to include at least three vehicles in the exhibit, and the three least expensive cars, the sixth, third, and second vehicle, cost 3,200 + 3,800 + 4,000 = \$11,000. On the other hand, exhibiting all vehicles costs \$26,700, so that any budget at or above this level will enable the organizer to exhibit all vehicles. Also notice the "granularity" of the solutions: an increase in the budget by \$1,000 results in an increase in the objective value by 7 (if increasing the budget from \$11,000 to \$12,000), by 14 (if the budget is increased from \$12,000 to \$13,000), by 7 if the budget is increased from \$13,000 to \$14,000, and so forth.

Table 4.13: Solutions of the car exhibit problem for different budgets

Budget (in \$10,000)	11	12	13	14	15	16	17	18	19
Cars included	2, 3, 6	2, 3, 4	1, 3, 6	1, 3, 4	1, 5, 6	1, 3, 5	1, 2, 3, 6	1, 2, 3, 4	1, 3, 5, 6
z-value	112	119	133	140	146	155	170	177	188

Budget	20	21	22	23	24	25	26	27
Cars included	1, 3, 4, 5	2, 3, 4, 5, 6	1, 2, 3, 4, 6	1, 3, 4, 5, 6	1, 2, 3, 4, 5	1, 2, 3, 4, 5	1, 2, 3, 4, 5	1, 2, 3, 4, 5, 6
z-value	195	207	210	228	232	232	232	265

Problem 6 (solving a problem via branch-and-bound): Consider the following all-integer optimization problem:

$$\begin{aligned} \text{P: Max} \quad & z = 7y_1 + 7y_2 \\ \text{s.t.} \quad & 6y_1 + 7y_2 \leq 34 \\ & 10y_1 + 6y_2 \leq 43 \\ & y_1, \; y_2 \geq 0 \text{ and integer.} \end{aligned}$$

(a) Produce the solution trees for branching starting with y_1 and y_2, respectively.
(b) What would the optimal solution be, if the integrality requirement for y_2 had been dropped?
(c) What are the additional constraints that were introduced between the root of the tree and the last integer solution found in the tree, assuming that branching starts with y_1?
(d) What would have happened, if the left child that resulted from the first branching had an objective value of 34.8?

Solution: (a) The solution that starts branching on y_1 is shown in Figure 4.13.

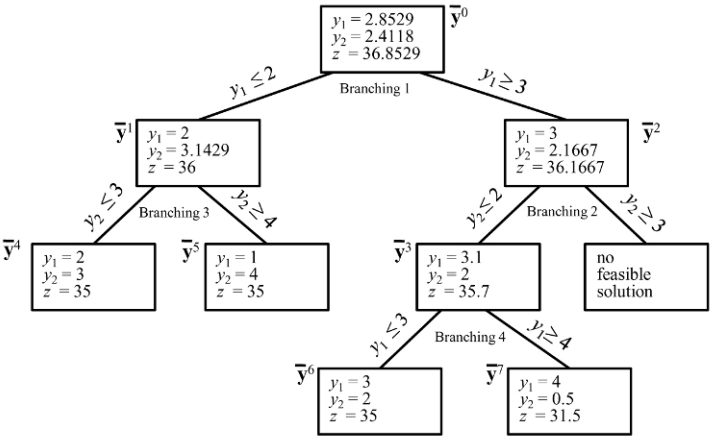


Figure 4.13

The solution tree that starts branching on y_2 is shown in Figure 4.14.

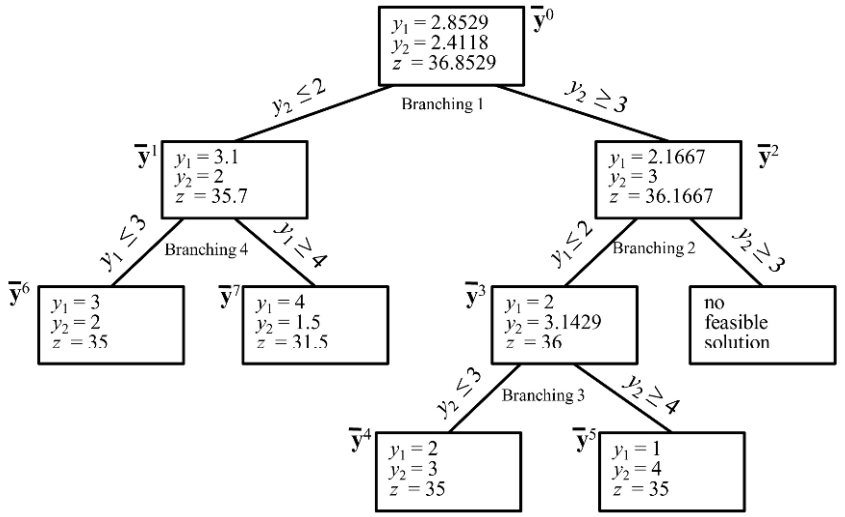


Figure 4.14

The problem has actually three alternative optimal solutions. Note that in both trees, branching 4 is necessary to complete the solution tree. The optimal solutions

are: $\bar{y}_1 = 3$ and $\bar{y}_2 = 2$, $\bar{y}_1 = 2$ and $\bar{y}_2 = 3$, as well as $\bar{y}_1 = 1$ and $\bar{y}_2 = 4$, all with a value of the objective function of $\bar{z} = 35$.

(b) In this case we would have had to start branching with y_1, so that Figure 4.13 applies. The optimal solution would be the right child of the root of the tree, i.e., solution $\bar{\mathbf{y}}^2$, which has $\bar{y}_1 = 3$ and $\bar{y}_2 = 2.1667$ with an objective value of $\bar{z} = 36.1667$.
(c) The additional constraints are $y_1 \geq 3$, $y_2 \leq 2$, and $y_1 \leq 3$.
(d) In both trees, we would never have branched from the left child of the first branching.

Problem 7 (choosing the correct branch-and-bound tree): Consider the following all-integer programming problem:

$$
\begin{aligned}
\text{P: Max } z = \; & 2y_1 + \; y_2 \\
\text{s.t.} \quad & -3y_1 + 15y_2 \leq 45 \\
& 3y_1 - \; 4y_2 \leq \; 9 \\
& y_1, \quad y_2 \geq \; 0 \text{ and integer.}
\end{aligned}
$$

Four solution trees have been developed by four different individuals, each claiming that their tree is correct. The trees are shown in Figures 4.15a, b, c, and d. However, only one of the trees is correct. Which one? For each solution tree, write one sentence that explains why this is or is not the correct tree.

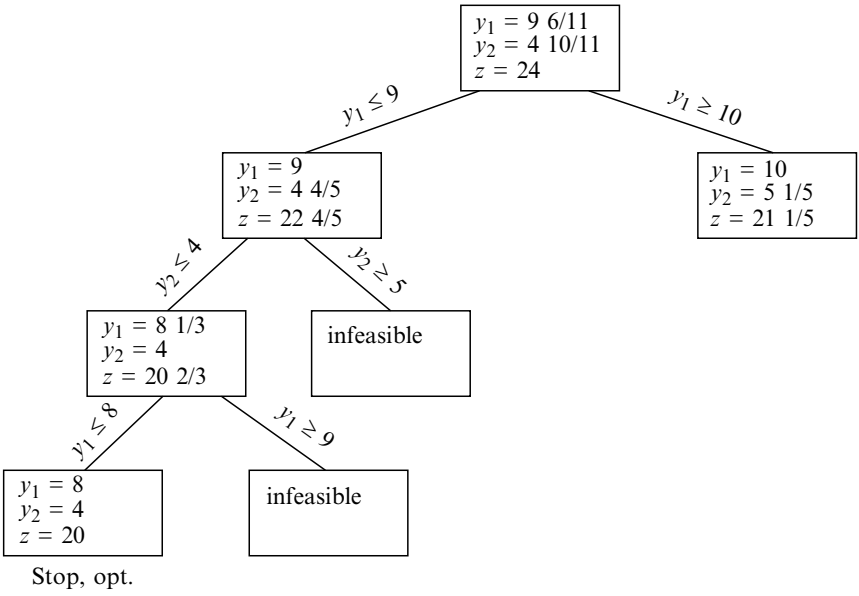


Figure 4.15a

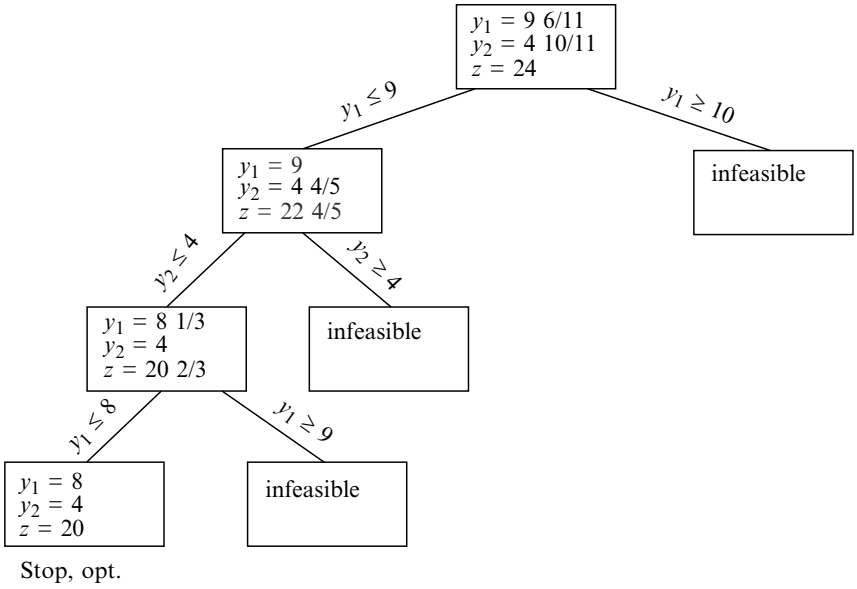


Figure 4.15b

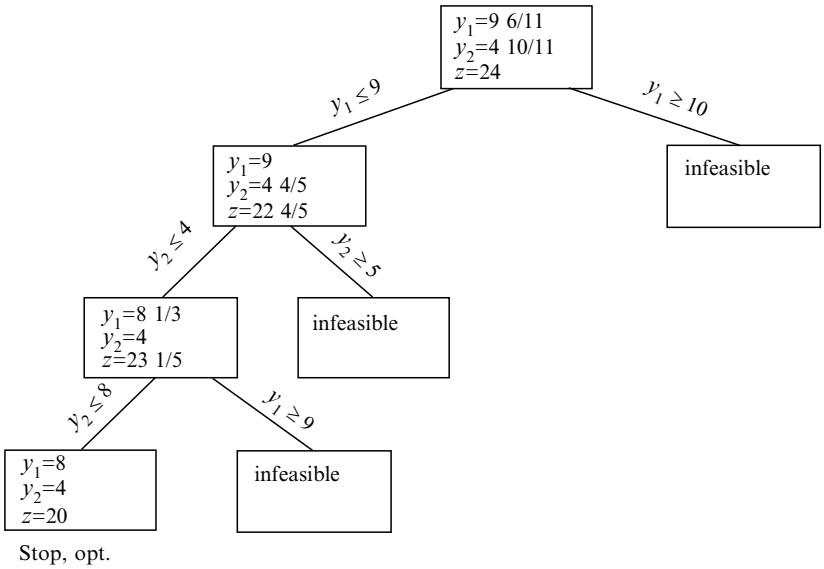


Figure 4.15c

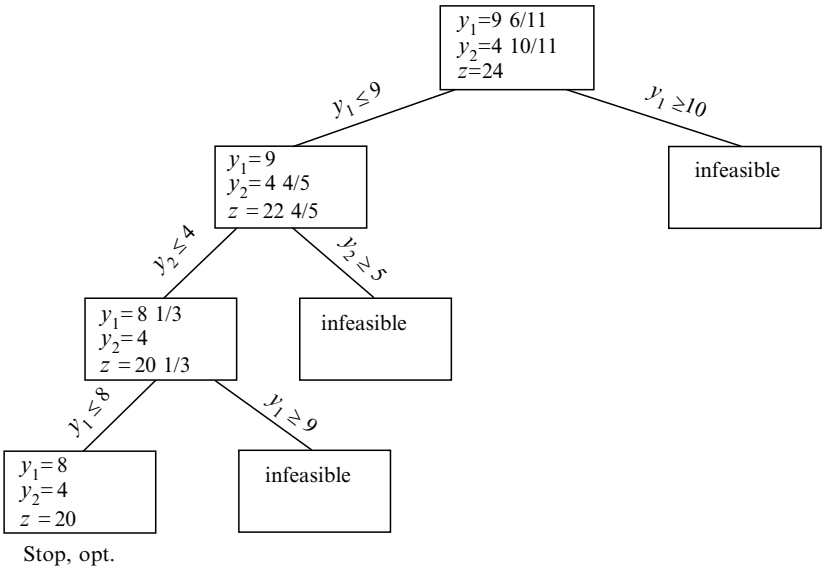


Figure 4.15d

Solution:

The solution tree in Figure 4.15a is false. The right child of the root of the tree is not feasible. If the solution there had been correct, branching should have continued at that node.

The solution tree in Figure 4.15b is false. The branching on the second level should be $y_2 \leq 4$ (as is), but the branch to the right child should be $y_2 \geq 5$, not $y_2 \geq 4$.

The solution tree in Figure 4.15c is false. The second branching leading to the left child has the objective value increase from $z = 22^4/_5$ to $z = 23^1/_3$, which cannot happen in a maximization problem.

The solution tree in Figure 4.15d is correct.

Problem 8 (heuristics: Greedy and Swap method): Consider the following integer programming problem.

$$\begin{aligned} \text{P: Max } z = {} & 21y_1 + 11y_2 + 65y_3 + 58y_4 + 122y_5 \\ \text{s.t.} \quad & 21y_1 + 10y_2 + 42y_3 + 37y_4 + 64y_5 \leq 640 \\ & y_2 \leq 2,\ y_3 \geq 1,\ 1 \leq y_4 \leq 2,\ y_5 \leq 2 \\ & y_1, \quad y_2, \quad y_3, \quad y_4, \quad y_5 \geq 0 \text{ and integer.} \end{aligned}$$

Use the Greedy algorithm and a Swap interchange to find a solution.

Solution: Ordering the variables with respect to their objective function contribution per resource unit, the order is y_5, followed by y_4, y_3, y_2, and y_1. Before we start increasing the values of the variables, we need to set the variables to their

minimum values, i.e. $y_3 = 1$ and $y_4 = 1$, so that we do not have to worry about lower bounds anymore. This leaves us with 640 – 42 – 37 = 561 resource units.

We now start the allocation with y_5. The upper bound is 2, so we set $y_5 = 2$, which leaves us with 561 – 2(64) = 433 resource units. The next best variable is y_4. As its upper bound equals 2 and its value already equals 1, we can only increase y_4 by 1. This leaves 433 – 1(37) = 396 resource units. The next valuable variable is y_3. It does not have an upper bound, so that we increase its value as much as the remaining resource units allow. We have 396 units left, each unit of y_3 requires 42 units, so that the largest value of $y_3 = 9$. Increasing y_3 by that value leaves us 396 – 9(42) = 18 resource units left. The next most valuable variable is y_2, whose upper bound is 2. However, the remaining resource units are only good for an increase of 1. This leaves us 8 resource units, which are not sufficient for any other increase. In summary, we have the solution $\mathbf{y} = [0, 1, 10, 2, 2]$, for which the objective value $z = 1{,}021$ may be calculated.

In the Swap procedure we will decrease the value of a variable by one, thus freeing some resources, which we then try to use by increasing the value of some other variable. For instance, decreasing the value of y_2 by one frees 10 units for a total of 18, which is not sufficient to increase any other variable by an integer amount.

Reducing the variable y_3 by frees up 42 units, so that 42 + 8 = 50 resource units are now available. Note that it also reduces the objective value by 65. Those resource units may be used to increase the value of y_1 by 2, which increases the objective value by 42, not enough to make up for the loss of 65. Alternatively, we may increase y_4 by one, which increases the objective value by 58, also not sufficient to make up for the loss.

We may now try to reduce the value of y_4 by 1, freeing 37 resource units for a total of 45. Note that the objective value decreases by 58 in the process. The resource may now be used to increase y_3 by one unit, which increases the value of the objective function by 65. This represents a net gain of +7, so that we make this change permanent. The new solution is now $\mathbf{y} = [0, 1, 11, 1, 2]$ with an objective value of $z = 1{,}028$. Three resource units remain available.

The process would continue here. We terminate the procedure at this point. It so happens that the solution found here is optimal.

5 Network Models

Graph theory, the subject at the root of this chapter, dates back to 1736, when the Swiss mathematician Leonard Euler considered the now famed "Königsberg bridge problem." At that time, there were seven bridges across the River Pregel that ran through the city of Königsberg on the Baltic Sea, and Euler wondered whether or not it would be possible to start somewhere in the city, walk across each of the bridges exactly once, and return to where he came from. (It was not). We will return to Euler's problem in Section 5.5 of this chapter. Two hundred years later in 1936, the Hungarian mathematician Denès König wrote the seminal book "The Theory of Finite and Infinite Graphs," that laid the foundations of modern graph theory. The subject was first used by operations researchers in the 1950s, most prominently by L.R. Ford and D.R. Fulkerson.

5.1 Definitions and Conventions

The models discussed in this section are optimization problems on a structure commonly known as a graph. A *graph* (for simplicity, we will refer to graphs also as *networks*, even though many graph-theorists will disagree) consists of *nodes* (sometimes referred to as *vertices*) and *arcs* (or *edges*). While many authors refer to undirected connections as edges and directed connections as arcs, we will make no such distinction here. The graph in Figure 5.1 is an example with the nodes n_1, n_2, n_3, n_4, and n_5 represented by circles and the arcs represented by directed or undirected lines. Arcs are written as either a_{ij} or (n_i, n_j), whatever is more convenient. A graph that contains only undirected edges is called an *undirected graph*, one with only directed arcs is a *directed graph* (frequently referred to as a network), and a graph that includes directed and undirected arcs is called a *mixed graph*.

H.A. Eiselt and C.-L. Sandblom, *Operations Research: A Model-Based Approach*,
DOI 10.1007/978-3-642-10326-1_5, © Springer-Verlag Berlin Heidelberg 2010

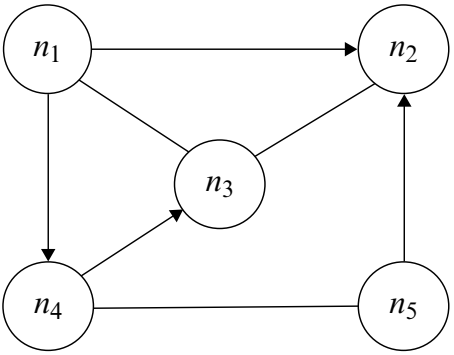


Figure 5.1

A *path* is defined as a sequence of nodes and arcs that starts at a node and ends at some node (possibly, but not necessarily, where it started). In the graph in Figure 5.1, the sequence $n_5 - a_{52} - n_2 - a_{23} - n_3$ is a path. A *cycle* in a graph is defined as a path that begins and starts at the same node. In the graph in Figure 5.1, the sequence $n_1 - a_{14} - n_4 - a_{43} - n_3 - a_{31} - n_1$ is a cycle. A node n_i is said to be *reachable* from another node n_i, if there exists at least one path from n_i to n_j. A *connected graph* is a graph, in which each node is reachable from each other node. The graph in Figure 5.1 is connected. A *tree* (or *tree graph*) is defined as a connected undirected graph, with the property that the removal of any arc or edge from it will render the graph disconnected. In that sense, a tree graph is the minimalist connected structure. An example for a tree is shown in Figure 5.2. Also note that there exists exactly one path between any pair of nodes in a tree.

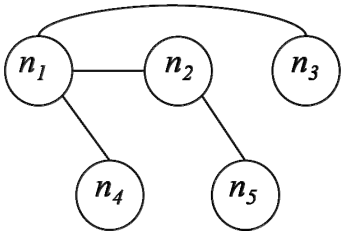


Figure 5.2

Any other problem-specific notation will be introduced whenever it is needed. We should mention that while all problems discussed in this chapter can be formulated as integer programming problems and solved with any of the pertinent solvers, such a process is typically inefficient and may hide internal structures that allow the user to gain insight into the problem. Typically, specific network-based algorithms are very efficient and outperform general-purpose integer programming solvers. However, such specific methods depend on the network structure. Even a single additional constraint may destroy the particular structure and will no longer allow

us to use the special algorithm, so that we have to return to standard integer programming formulations.

5.2 Network Flow Problems

In its simplest form, a network flow problem can be described as follows. It starts with a graph, in which an unlimited (or, at least, sufficiently large) quantity of a homogeneous good is available at a node called the *source*, while as many units as possible are to be shipped from the source through the network to another node, named the *sink*, where these units are in demand. (Some authors call this an *O – D flow*, which refers to the flow from origin to destination). Units can be shipped through as many nodes as necessary, as long as they do not exceed the *capacity* specified at each of the arcs in the network. It is assumed that there are no losses anywhere in the network. Once the problem has been solved, the optimal *flow pattern* indicates how many units are shipped on which arcs, and the optimal *flow value* indicates how many flow units are shipped through the network.

One of the many examples deals with the evacuation of people from buildings, convention halls, or even cities. As an example, consider the situation shown in Figure 5.3.

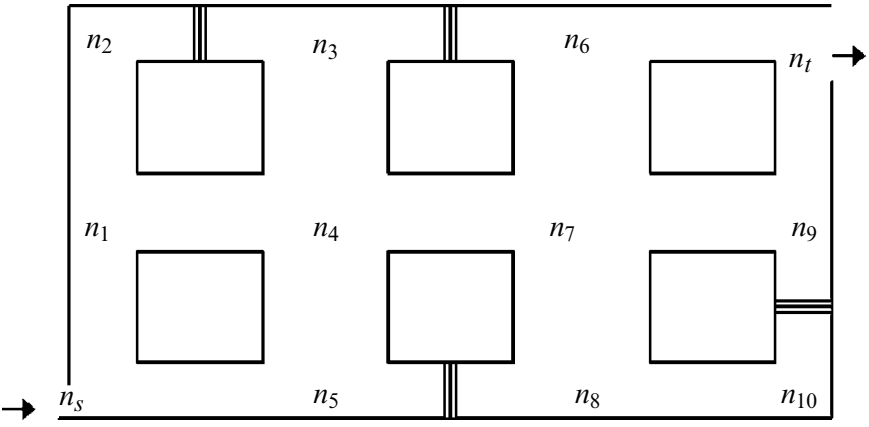


Figure 5.3

The floor plan depicts a large exhibition hall with the squares denoting kiosks. An escape plan is needed for the auditorium on the left that has only one entrance through the exhibit hall at the bottom left, labeled by n_s at the bottom left of Figure 5.3. From n_s, people have to be evacuated through the exhibit hall towards the door to the outside, marked by n_t on the top right of Figure 5.3. The walkways along the walls of the exhibit hall are fairly wide, and each of their segments (from one corner to another) was shown to let up to 60 people per minute through

in an emergency situation. The walkways in the center are narrower, so that only 40 people per minute can pass along each segment. The steps on some of the outside segments shown by lines across the walkway, present additional obstacles, so that these segments only allow 30 people per minute to pass through them. The question is now how many people can pass through the exhibition hall in each minute. Given the capacity of the auditorium, it can then be computed how long an evacuation of the auditorium through the exhibit hall (from n_s to n_t) will take. The graph in Figure 5.4 shows an image of the situation with the appropriate arc capacities. The quest is now to find the largest possible flow from n_s to n_t.

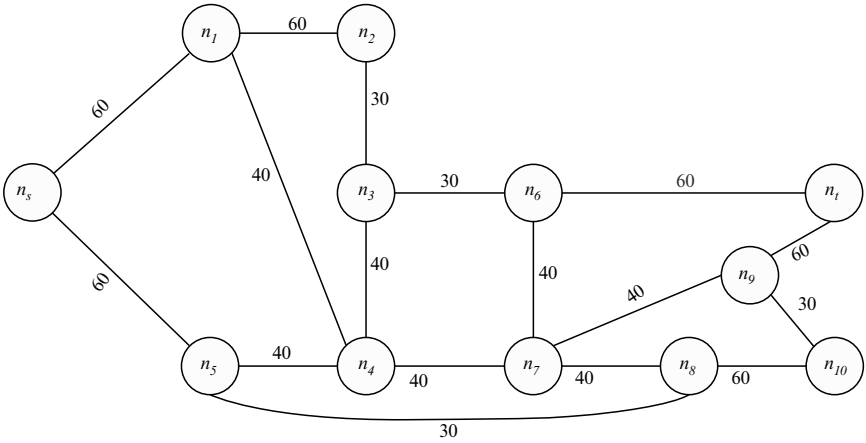


Figure 5.4

A number of additional applications of flow problems can be found in Eiselt and Sandblom (2000).

In order to formulate the problem, we first need to define variables. Here, the continuous flow variables are x_{ij}, defined as the flow along the arc a_{ij} from n_i directly to node n_j. Consider the example of Figure 5.4. Since all edges are undirected, we need to define two variables for each edge, one for each direction. Given that there are 17 edges, we will need 34 variables. (Some of the variables can be ignored, though: we do not have to consider flows into the source, as the entire purpose is to get as much flow out of there as possible. In other words, we do not need to define x_{1s} and x_{5s}. Similarly, we do not need to consider flows out of the sink). The objective will then maximize the flow through the network. This flow can be expressed as the number of flow units that are sent out of the source. In the example of Figure 5.4, this is $x_{s1} + x_{s5}$. Alternatively, we can consider the number of flow units that are sent into the sink, i.e., $x_{6t} + x_{9t}$.

As far as constraints are concerned, we have to consider two types. The first set of constraints will have to ensure that no flow is lost at any of the nodes. Such

constraints are commonly called *conservation equations*, (*flow*) *balancing equations*, or *Kirchhoff node equations* in reference to their counterpart in electrical networks. They are formulated by requiring that the number of units that flow into a node equals the number of units that flow out of a node. In the above example, the conservation equation for the node n_3 is $x_{23} + x_{43} + x_{63} = x_{32} + x_{34} + x_{36}$. Conservation equations have to be formulated for each node except for the source and the sink. The second set of constraints are the capacity constraints that require that the flow along each segment does not exceed the upper limit. Again, in this example we have $x_{s1} \leq 60$, $x_{s5} \leq 60$, $x_{12} \leq 60$, $x_{21} \leq 60$, and so forth. Adding the nonegativity constraints, a mathematical formulation has been obtained and the problem could be solved with any off-the-shelf linear programming software. Whenever available, specialized algorithms are, however, much faster, as they can use the special structure of the problem.

Ford and Fulkerson were the first to describe a method to solve the maximal flow problem. Their technique belongs to the large class of so-called *labeling methods*. Labeling methods have been developed for many network models, and they all have in common that they are very efficient. The idea of Ford and Fulkerson's incremental method is to (incrementally) increase the flow in forward arcs, and to decrease the flow in backward arcs. Both of these steps are necessary in order to reach an optimal solution. In order to illustrate, consider a "forward arc," in which we would like to increase the flow. If the capacity of such an arc is, say, 7, and the present flow in the arc is 4, then we are able to increase the flow on this arc by the present slack of $7 - 4 = 3$ units. On the other hand, if a "backward arc" has a capacity of 7 and a present flow of 4, we can decrease the flow on this arc by no more than its present flow, i.e., by 4 units. This is the fundamental ideal of the labeling technique. Starting with the source, the method attempts to label nodes along arcs. The first part of a node's label indicate the neighboring node it was labeled from, while the second part of the label indicates the possible flow change on the arc along which the labeling took place.

As in all procedures based on dynamic programming, the objective value—the second part of the label—is determined by the objective value achieved so far plus whatever contribution to the objective function has to be accounted for in this step. More specifically, labeling a node n_j from a node n_i along an arc a_{ij} means that the incremental flow that can be squeezed from the source to n_j equals the minimum of what incremental flow can be sent from the source to n_i plus whatever slack capacity we have along the arc a_{ij}.

Once all nodes that can be labeled have been labeled, the iteration comes to an end in one of two states. We either have been able to label the sink, in which case a *breakthrough* has occurred, or we were not able to label the sink, which is referred to as a *nonbreakthrough*. If a breakthrough has occurred, we are able to increase the flow through a network, while in case of a nonbreakthrough, the current flow is maximal and the procedure terminates.

The algorithm is best explained by means of a numerical

Example: Consider the network in Figure 5.5, whose numbers next to the arcs indicate the capacity of the arc.

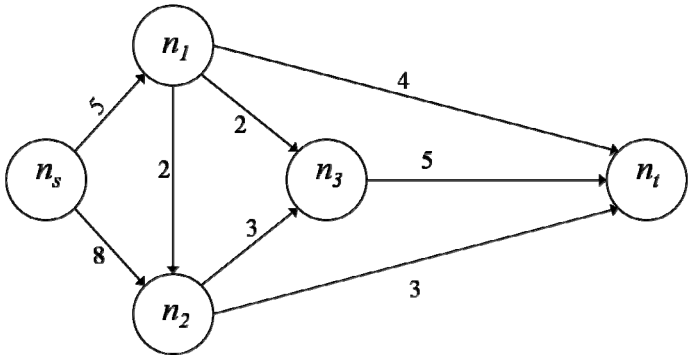


Figure 5.5

Suppose that the source n_s is now labeled with (n_s, ∞), indicating that we start at the node n_s, and so far, we can change the flow by any amount we choose. At this point we could label either n_1 and/or n_2. Suppose we choose n_1. Since we label the node n_1 from n_s, the arc (n_s, n_1) is a forward arc and we attempt to increase its flow. The capacity of the arc is 5, while its present flow is zero, meaning that we can increase the flow in this arc by 5 units. This is indicated in the label of n_1, which is then $(n_s, 5)$. Now the nodes n_s and n_1 are labeled, and the process continues. At this point, we have a large number of choices: we can either label n_2 from the source n_s, or label n_2, n_3 or n_t from n_1. Any technique that labels all nodes from one node before moving on is referred to as *breadth-first-search*, while a labeling strategy that attempts to move on a quickly as possible is called *depth-first-search*. Suppose that we choose to label n_2 from n_1. Again, we are moving with the direction of the arrow, so that this is a forward labeling step. The label of the node n_2 will then be $(n_1, \min\{5, 2-0\}) = (n_1, 2)$. The reason is that while we could increase the flow from the source on some path (we don't have to know at this point which path) to n_1 by 5 units, we can only ship two more units from n_1 to n_2. At this point, n_2 has been labeled and one of the many choices to continue labeling is to label the sink n_t from n_2. The label of the sink n_t is then $(n_2, \min\{2, 3-0\}) = (n_2, 2)$. Now the sink has been labeled, and we have achieved a breakthrough.

Whenever a breakthrough occurs, we are able to increase the flow through the network by at least one unit. In order to do so, we now have to retrieve the path on which the flow change is possible. This is where the first part of the labels comes in. In a backward recursion, we start at the sink n_t. Its label indicates that its predecessor is n_2. Now the label of n_2 shows that its predecessor was n_1, whose

predecessor, in turn, was n_s. This means that we have successfully retrieved the path $n_s - n_1 - n_2 - n_t$. This is the path on which the flow will be increased by 2 units, which is the second part of the label of the sink. The resulting flow pattern is shown in Figure 5.6, where the arcs have two values: its capacity, and its present flow.

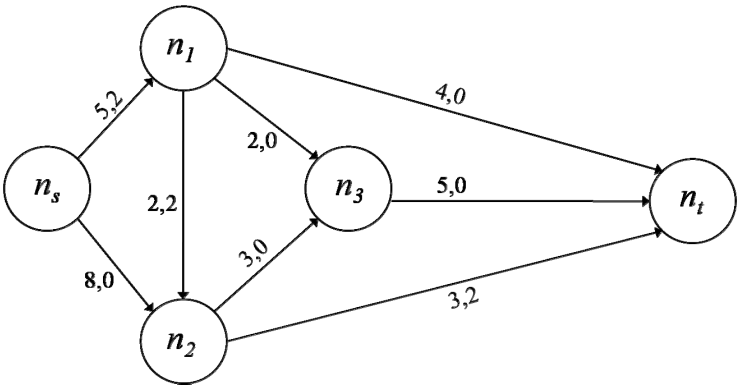


Figure 5.6

We now delete all labels except the label of the source, and start anew. We can again label n_1 from the source n_s, this time the label of n_1 is $(n_s, \min\{\infty, 5-2\}) = (n_s, 3)$. From n_1, it is possible to label the sink, whose label is then $(n_1, \min\{3, 4-0\}) = (n_1, 3)$. Again, we have obtained a breakthrough and the flow can be increased by 3 units. The path on which this increase takes place is determined by following the labels backwards from the sinks, which results in $n_s - n_1 - n_t$. The resulting flow pattern is shown in Figure 5.7.

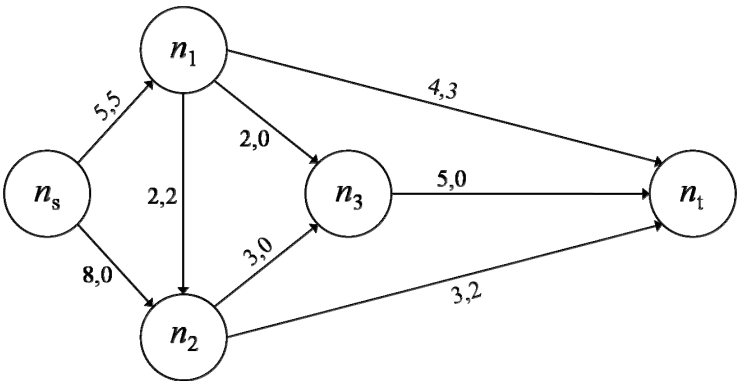


Figure 5.7

Again, resetting the labels, the process begins again. At this point, it is no longer possible to label the node n_1 from the source, as the flow has reached the capacity. However, we can still label the node n_2 with $(n_s, \min\{\infty, 8–0\}) = (n_s, 8)$. From n_2, we can label the sink n_t with the label $(n_2, \min\{8, 3–2\}) = (n_2, 1)$, and we have achieved another breakthrough, on which the flow can be changed by one unit. The path can be retrieved as $n_s – n_2 – n_t$. The flow pattern is then shown in Figure 5.8.

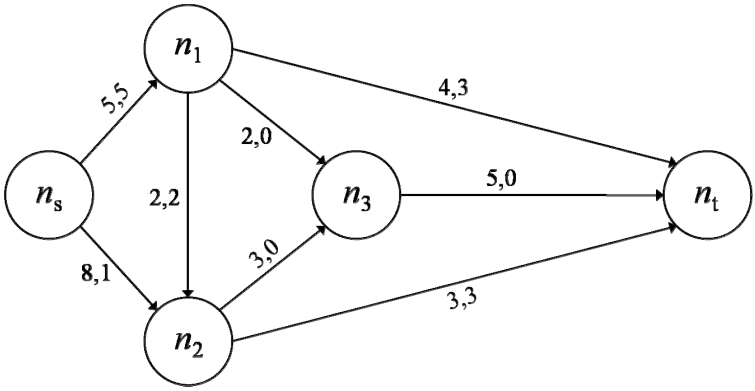


Figure 5.8

The next iteration again starts by labeling the node n_2 from the sink, which is again our only choice. The label of the node n_2 is now $(n_s, \min\{\infty, 8–1\}) = (n_s, 7)$. We can now label n_3 from n_2, so that the label on node n_3 is $(n_2, \min\{7, 3–0\}) = (n_2, 3)$. From n_3, we can then label the sink with $(n_3, \min\{3, 5–0\}) = (n_3, 3)$, and another breakthrough has occurred. The path on which the flow is changed can be retrieved as $n_s – n_2 – n_3 – n_t$, and on all arcs along that path the flow is increased by 3 units. The resulting flow pattern is then shown in Figure 5.9.

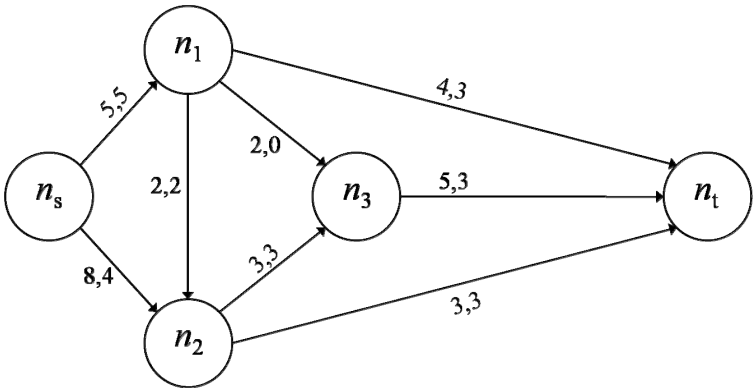


Figure 5.9

Notice that so far, we have only used the forward labeling in the incremental method. The next iteration commences by resetting the labels and restarting the process. Given that the source is labeled as usual with (n_s, ∞), the only choice is now to label the node n_2 with $(n_s, \min\{\infty, 8–4\}) = (n_s, 4)$. From the node n_2, no forward labeling is possible. However, we can follow the arc from n_1 to n_2 against the direction of the arc in a backward labeling step. This results in node n_1 receiving the label $(n_2, \min\{4, 2\}) = (n_2, 2)$. From node n_1, we can then either label n_t or n_3. We choose n_3, label it with $(n_1, \min\{2, 2–0\}) = (n_1, 2)$, and from n_3, we can then label the sink n_t with $(n_3, \min\{2, 5–3\}) = (n_3, 2)$. Another breakthrough has occurred, and the flow through the network can be increased by 2 units. The flow along which the flow will be changed is retrieved through backward recursion as $n_s – n_2 – n_1 – n_3 – n_t$, where the arc from n_1 to n_2 is used in reverse direction. the new flow pattern is determined by increasing the flow in all forward arcs on that path, while decreasing the flow in the solitary backward arc on the path. The resulting flow pattern is shown in Figure 5.10.

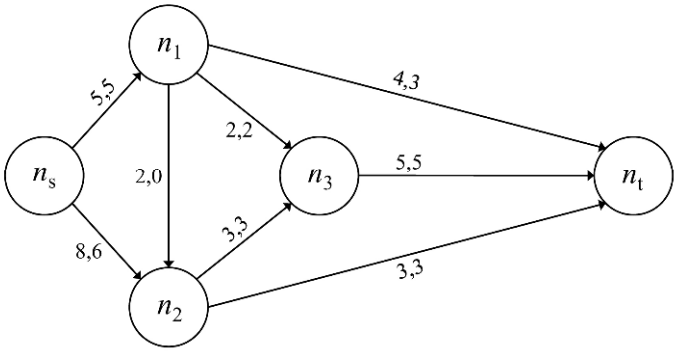


Figure 5.10

In the next step, we start again by labeling the node n_2 from the source, which receives the label $(n_s, \min\{\infty, 8–6\}) = (n_s, 2)$. At this point, further progress is blocked. The only unlabeled node adjacent to the source is n_1, and the arc (n_s, n_1) is filled to capacity. From the node n_2, further (forward) flow to n_3 and n_t cannot be sent, as both arc flows are at capacity. Also, labeling node n_1 from n_2 is not possible, as the arc flow on that (backward) arc is already at the lower bound of zero. At this point we have labeled all nodes that can be labeled, and we were not able to label the sink. This is a nonbreakthrough. This indicates that the present flow pattern is indeed a *maximal flow* with a total flow of 11 units, the number of flow units that leave the source and, since no flow is lost along the way, the number of units that arrive at the sink. We would like to point out that this maximal flow is not unique: sending one less unit from n_1 to n_3, and on to n_t and shipping it instead directly from n_1 to n_t results in a different flow pattern with the same flow value.

Assessing the situation, we find that we now have the set $N_s = \{n_s, n_2\}$ of labeled nodes, and its complement $N_t = \{n_1, n_3, n_t\}$ of unlabeled nodes. Observe that the flows of all arcs that lead from a node in N_s to a node in N_t (here: the arcs (n_s, n_1), (n_2, n_3), and (n_2, n_t)) are at capacity, while the flows of all arcs leading from a node in N_t to a node in N_s (here: the arc (n_1, n_2)) are at the zero level. All arcs that lead from a node in N_s to a node in N_t are said to be included in the *minimal cut C*. In our example, $C = \{(n_s, n_1), (n_2, n_3), (n_2, n_t)\}$. Adding the capacities (not flows) of all arcs in the minimal cut results in the value of the minimal cut, which, in or example, equals 5 + 3 + 3 = 11. This leads to the famous

Theorem (Ford and Fulkerson): The value of a maximal flow equals the value of a minimal cut.

The minimal cut constitutes a bottleneck in the network. If we want to increase the capacities of some arcs in the network so as to be able to increase the flow through the network, we have to increase the capacities of arcs that are in the minimal cut(s). (Note that the minimal cut is not necessarily unique. As an example, consider the network in Figure 5.11 with capacities next to the arcs. The broken lines refer to the cuts $C_1 = \{(n_s, n_1), (n_s, n_2)\}$, $C_2 = \{(n_s, n_1), (n_2, n_t)\}$, and $C_3 = \{(n_1, n_t), (n_2, n_t)\}$. All cuts have a capacity equal to 5. Note that in case of multiple cuts, the method described above will find only the minimal cut that is closest to the source.

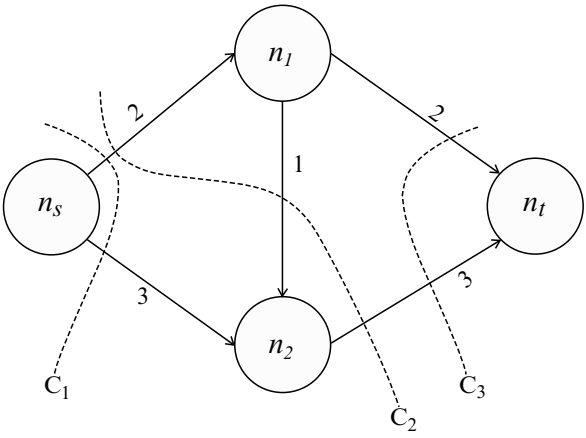


Figure 5.11

A variety of extensions of the maximal flow problem exists. One of the most popular generalization is the *min-cost feasible flow problem*. Again, the idea is very simple. In a network with a designated source and sink, each arc a_{ij} has a

lower bound λ_{ij} and an upper bound κ_{ij} on the flow. (Recall that in the above max flow problem all lower bounds were assumed to be zero). In addition, it is assumed to cost c_{ij} dollars to send one unit of flow on the arc a_{ij}. The problem is now to find a flow from source to sink that respects all lower and upper bounds on the flows and that minimizes the total shipping costs.

As an example, consider the graph in Figure 5.12, where the numbers next to the arcs symbolize the lower bounds λ_{ij}, the upper bounds κ_{ij}, and the unit costs c_{ij}.

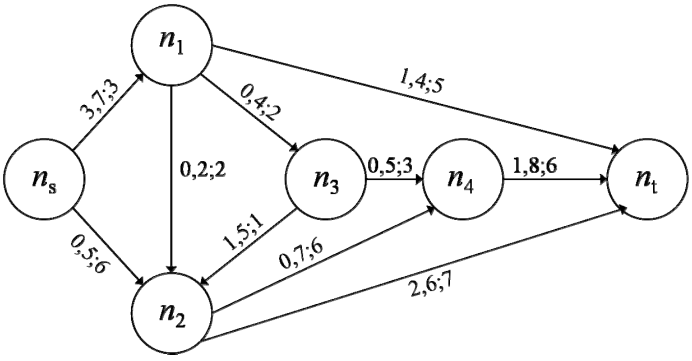


Figure 5.12

The cost-minimizing flow problem has three types of constraints: the usual conservation equations, along with the lower and upper bounds on each arc flow. This specific problem can be formulated as follows.

$$\text{Min } z = 3x_{s1} + 6x_{s2} + 2x_{12} + 2x_{13} + 5x_{1t} + 6x_{24} + 7x_{2t} + 1x_{32} + 3x_{34} + 6x_{4t}$$

$$\text{s.t. } x_{s1} - x_{12} - x_{13} - x_{1t} = 0$$
$$x_{s2} + x_{12} + x_{32} - x_{24} - x_{2t} = 0$$
$$x_{13} - x_{32} - x_{34} = 0$$
$$x_{34} + x_{24} - x_{4t} = 0$$
$$x_{s1} \geq 3 \qquad x_{13} \leq 4$$
$$x_{32} \geq 1 \qquad x_{32} \leq 5$$
$$x_{1t} \geq 1 \qquad x_{1t} \leq 4$$
$$x_{2t} \geq 2 \qquad x_{34} \leq 5$$
$$x_{4t} \geq 1 \qquad x_{24} \leq 7$$
$$x_{s1} \leq 7 \qquad x_{2t} \leq 6$$
$$x_{s2} \leq 5 \qquad x_{4t} \leq 8$$
$$x_{12} \leq 2$$

$x_{ij} \geq 0$ for all arcs i, j.

The solution can be obtained by any standard optimization package or specialized algorithm. The optimal flow pattern is shown in Figure 5.13, the total flow from source to sink is 4, and the associated total transportation costs are 47.

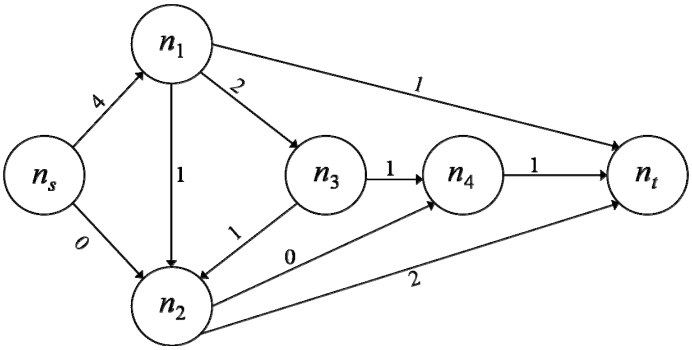


Figure 5.13

Further extensions are possible. One such extension has a required flow value $\bar{f}$. A typical network modification that accomplishes this requirement will simply add an arc (n_t, n_s) with zero costs and lower and upper bound equal to $\bar{f}$. The original flow problem has then be reformulated as a circulation, in which the conservation equations have to hold for all nodes, including the source and the sink. Working with the mathematical formulation, all we have to do is add a single constraint that requires that the flow value equals $\bar{f}$. If $\bar{f} = 7$ in the above example, we either add the constraint $x_{s1} + x_{s2} = 7$, or, alternatively, $x_{1t} + x_{2t} + x_{4t} = 7$ and re-solve. (In the above numerical example, the new solution will be the same as that shown in Figure 5.13, except that an additional 3 flow units are shipped from n_s to n_1 and on to n_t).

It is now also possible to demonstrate how to cast the standard transportation problem (see Section 2.2) into the mold of cost-minimal network flow problems. In terms of the network, we will make the following modifications. In addition to the existing origins and destinations, create an artificial source n_s and artificial sink n_t. Connect the source with all origins, and connect all destinations with the sink. (This type of problem reformulation can be used for all problems with multiple sources and sinks). The lower bounds of all (source, origin) connections are zero, while the upper bounds equal the supplies available at the respective origin. Similarly, all (destination, sink) arcs have a zero lower bound and an upper bound that equals the demand at the respective destination. Both types of arcs have zero costs. The existing arcs that connect the origins and the destinations have zero lower bounds and arbitrarily large upper bounds (except in cases, in which capacities need to be considered), and carry the costs specified for the original problem. Note that so far, the zero flow would be optimal, as none of the arcs requires a flow greater than zero, and any flow from source to sink costs

money. In order to force flow through the network, we connect the sink with the source by means of an artificial arc (n_t, n_s) that has zero costs, and an upper and lower bound that are both equal to the minimum of the total supply and the total demand at the sources and destinations, respectively. This way, the (sink, source) arc forces as many flow units through the network as are in demand or are needed, whatever is less.

Another possible extension includes capacity constraints at the nodes. Two approaches are possible. The first uses the given network and modifies it so that it includes node capacities. This can be accomplished by "splitting" all of the nodes with node capacities. In particular, a node n_i with node capacity κ_i is then replaced by an "in-node" n_i' into which all arcs lead that led into the original node n_i, an "out-node" n_i'', out of which all arcs lead, that lead out of the original node n_i. Finally, the two new nodes n_i' and n_i'' are connected by an arc (n_i', n_i''), whose arc capacity is the original node capacity κ_i. What we have done in this approach is simply to replace the node capacity by an arc capacity.

An alternative approach simply uses a mathematical programming formulation. The number of flow units that flow through a node equals the number of units that enter (and, as nothing is lost, leave) a node, and an appropriate additional constraint is added. In the min-cost flow example of Figure 5.12 a capacity of, say, 5 units through node n_2 can be written as $x_{s2} + x_{12} + x_{32} \leq 5$, or, equivalently, by using the outflow, as $x_{24} + x_{2t} \leq 5$.

5.3 Shortest Path Problems

Similar to the maximum flow problem discussed in the previous section, shortest path problems are easily described. Given a network with a prespecified source and sink node as well as arc values c_{ij} that denote the cost (or distance, fuel, or any other disutility) of traveling from node n_i directly to node n_j along arc a_{ij}, the task is to find the shortest path from the source to the sink. The literature typically distinguishes between one-to-one shortest path problems (those that search the shortest path between source and sink), one-to-all shortest path problems (in which the task is to find the shortest paths between the source and all other nodes in the network), and the all-to-all shortest path problems (where the task is to determine the shortest paths between all pairs of nodes). Clearly, it would be possible—albeit inefficient—to use an algorithm for the one-to-one shortest path problem and apply it repeatedly so as to solve the one-to-all and all-to-all shortest path problems.

This section first describes a way to reformulate the one-to-one shortest path problem, so that it fits into the mold of the cost-min feasible flow problem. It then describes the workings of an all-to-all shortest path algorithm that is not only very

efficient, but also needed in areas such as location models, where all shortest paths have to be known before any location algorithm can even start.

First, we will discuss how to reformulate a shortest path problem as a min cost flow problem. The idea is to force one flow unit through the network, and let the optimizer find the cost-minimal, i.e., shortest, path. This is done by having lower bounds of zero and upper bounds of one for all arcs, coupled with the actual costs or distances specified for all arcs. So far, the optimal solution would be the zero flow, as it is feasible and no cheaper solution can exist (at least not as long as the arc distances are nonnegative). We then add the circulatory arc (n_t, n_s) with lower and upper bound equal to one. This forces a single unit through the network and will result in the desired solution. The graph transformation is shown in Figure 5.14, where Figure 5.14a shows the original graph with the distances or costs next to the arcs, while Figure 5.14b has the lower bounds, the upper bounds, and the cost/distances at the arcs.

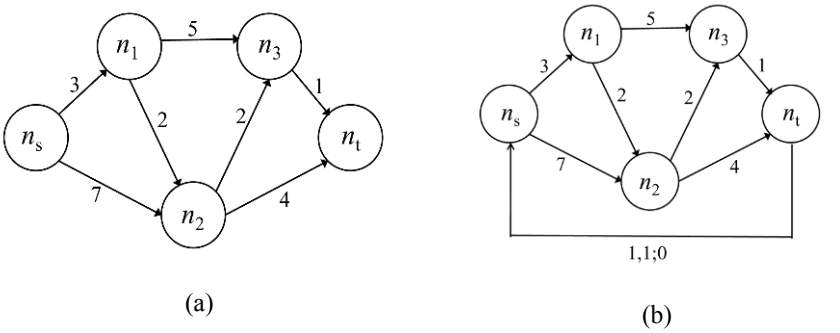


Figure 5.14

As far as the mathematical formulation is concerned, we can use the usual min-cost objective function, coupled with two sets of constraints. The first single constraint ensures that exactly one unit leaves the source. The second set of constraint are the usual conservation equations, which ensure that the flow unit that leaves the source has only one place to go: the sink. In the example of Figure 5.14, the formulation would be as follows:

$$\text{Min } z = 3x_{s1} + 7x_{s2} + 5x_{13} + 2x_{12} + 2x_{23} + 1x_{3t} + 4x_{2t}$$
$$\text{s.t. } x_{s1} + x_{s2} = 1$$
$$x_{s1} - x_{12} - x_{13} = 0$$
$$x_{s2} + x_{12} - x_{23} - x_{2t} = 0$$
$$x_{13} + x_{23} - x_{3t} = 0$$
$$x_{ij} \geq 0 \text{ for all } i, j.$$

Alternatively, the first constraint $x_{s1} + x_{s2} = 1$ could be replaced by the constraint $x_{2t} + x_{3t} = 1$, representing the flow into the sink.

Shortest path problems have many real-world applications. In addition to the obvious applications, in which the shortest path in a road network is to be found (e.g., for GPS-based navigation systems), shortest path problems occur in scenarios that seemingly have nothing to do with shortest paths. Instead, a process has "discretized," i.e., subdivided into a finite number of states that described the system at that point in time. Each node of the network symbolizes a state of the system, and an arc indicates a possible transition from one state to another. The arc values show the amount of resources, such as time, money, or fuel, that a transition takes.

A good example of such a problem deals with the problem of getting an aircraft from a standstill position to a certain speed and altitude in the fastest possible way. This problem is particularly relevant for fighter aircraft.

As a numerical example, consider the situation shown in Figure 5.15. The two numbers in each node indicate the state the aircraft is in: the first component is the ground speed in 100 mph, and the second number shows the aircraft's altitude in 1,000 ft. Initially, the aircraft is at (0, 0), i.e., standing still on the ground. Each nodes describes a state the aircraft can be in with respect to speed and altitude, and the arcs between these states indicate the possible transitions from one state to another. The values next to the arcs show the time (in seconds) that is required to make the transition from one node to another. For example, it takes five seconds to bring the aircraft from the standstill position at (0, 0) to (1, 0), i.e., a speed of 100 mph at zero altitude (meaning on the runway). Suppose that it is desired to bring the aircraft from a standstill position to a speed of 500 mph and an altitude of 4,000 ft as quickly as possible.

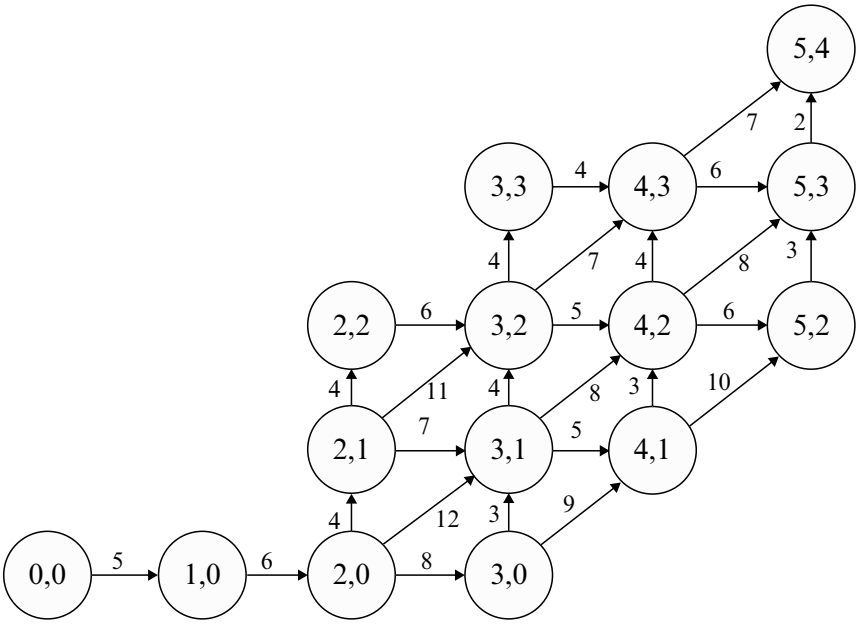


Figure 5.15

At each state in this example, the pilot has three options: either stay at the same altitude and speed up, remain at the same speed and climb, or speed up and climb simultaneously. The shortest path in the above example includes the nodes (0, 0), (1, 0), (2, 0), (2, 1), (2, 2), (3, 2), (4, 3), and (5, 4). In other words, the instructions to the pilot would state to bring the aircraft from a standstill position to a a speed of 200 mph, then remain at that speed and climb 2,000 ft, then stay at that altitude and speed up to 300 mph, and then accelerate and climb simultaneously to the desired speed of 500 mph and altitude of 4,000 ft. This way, it will take 39 seconds, the length of the shortest path, to reach the desired state.

One of the most popular techniques for the determination of shortest paths from one node to all other nodes is Dijkstra's technique that was first published in the late 1950s. It is a highly efficient method that belongs to the class of so-called label setting techniques (as opposed to other label-correcting techniques). The main idea is to label the nodes n_s, n_1, …, with labels $L(n_s)$, $L(n_1)$, …, so that each label consists of two parts. The first is the immediate predecessor of the node on the shortest path known so far, and the second part is the length of the shortest path known so far. Throughout the procedure, we distinguish between nodes that have a *temporary label* and those with a *permanent label*. The label of a permanently labeled node indicates the actual length of the shortest path from the source to this node as well as the immediate predecessor on that path, while the temporary label comprises the presently shortest known path and the node's immediate predecessor on it. In each step of the algorithm, one node with a

temporary label is chosen and its label made permanent. The upper bounds on the estimates of the shortest paths to all direct successors of this node are revised, and the process is repeated until the labels of all nodes are made permanent. It is important to realize that Dijkstra's method is only applicable to networks with nonnegative arc lengths.

To initialize the method, assume that the source n_s is labeled $L(n_s) = (n_s, 0)$, while all other nodes n_j are labeled $L(n_j) = (n_j, \infty)$. In the beginning, all nodes are assumed to be temporary. The method now proceeds as follows. We choose the temporarily labeled node whose (second part of the) label is minimal among all temporarily labeled nodes. Ties are broken arbitrarily. The label of this node is then made permanent. Suppose this node is n_i. All of this node's direct successors are then investigated by comparing their second part of the label with the label of n_i plus the arc length of the arc a_{ij}. If the preset label of n_j is smaller, we leave it unchanged; if it is larger, we replace it by setting $L(n_j) = (n_i, L(n_i) + c_{ij})$, i.e., by the label of n_i plus the length of the arc that connects n_i and n_j.

As an example of the procedure, consider the network in Figure 5.16.

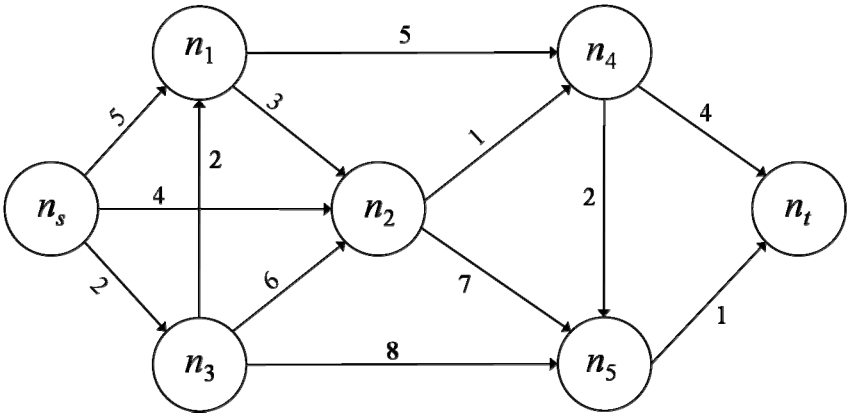


Figure 5.16

As indicated above, in the initialization step ("Step 0"), we label the source as $L(n_s) = (n_s, 0)$ and all other nodes n_j with $L(n_j) = (n_j, \infty)$, and let all nodes be temporary. The node with the lowest temporary label is the sink, so that it is chosen and its label is made permanent. All computations discussed here are summarized in Table 5.1, where the first time a label has been made permanent, it receives a "*" and due to its permanent status, it is not listed again below. Choosing n_s to receive a permanent label means that the labels of its direct successors n_1, n_2, and n_3 may have to be revised (the labels of all other nodes remain unchanged). Their present labels are compared with the label from n_s,

which is the source's label (here: 0) plus the length of the arc from n_s to the node in question. For n_1, the comparison is between ∞ and 0 + 5, which is 5, so that n_1 is now labeled from n_s, which is indicated in its new temporary label $L(n_1) = (n_s, 5)$. Similarly, the labels of the nodes n_2 and n_3 are $(n_s, 4)$ and $(n_s, 2)$, respectively.

We are now at the end of Step 1 in Table 5.1, where we choose the node with the lowest temporary label. In this example, the node is n_3, as it has the smallest label with "2." We now make this label permanent and revise the labels of its successors n_1, n_2, and n_5 in Step 2. The present label of the node n_1 indicates that a path of length 5 is already known from the source. If we were to label n_1 from n_3, its label would be 2 + 2 = 4, which is shorter, so that the new label of $L(n_1) = (n_3, 4)$. For n_2, we find that the presently shortest known path is of length 4 (its present label), while labeling the node from n_3 would lead us to a path of length of 2 + 6 = 8. Since this new path is longer, we ignore it and leave the label of n_2 unchanged. Finally in this step, the present label of n_5 indicates a path of length ∞ is known, which is compared to the label the node would receive if labeled from n_3, which is 2 + 8 = 12. This new label is shorter, so that node n_5 receives the new label $L(n_5) = (n_3, 10)$.

This process continues until all nodes have been permanently labeled. The intermediate and final results are shown in Table 5.1.

Table 5.1: Permanent and temporary labels during the Dijkstra method

Step #	$L(n_s)$	$L(n_1)$	$L(n_2)$	$L(n_3)$	$L(n_4)$	$L(n_5)$	$L(n_t)$
0	$(n_s, 0)$*	(n_1, ∞)	(n_2, ∞)	(n_3, ∞)	(n_4, ∞)	(n_5, ∞)	(n_t, ∞)
1		$(n_s, 5)$	$(n_s, 4)$	$(n_s, 2)$*	(n_4, ∞)	(n_5, ∞)	(n_t, ∞)
2		$(n_3, 4)$*	$(n_s, 4)$		(n_4, ∞)	$(n_3, 10)$	(n_t, ∞)
3			$(n_s, 4)$*		$(n_1, 9)$	$(n_3, 10)$	(n_t, ∞)
4					$(n_2, 5)$*	$(n_3, 10)$	(n_t, ∞)
5						$(n_4, 7)$*	$(n_4, 9)$
6							$(n_5, 8)$*

The results in Table 5.1 can now be used to determine the *tree of shortest paths rooted at* n_s. This is done by choosing all permanent labels and connect the node with its direct predecessor as specified in the label. In our example, the node n_t has n_5 as its direct predecessor, so we introduce the arc a_{5t}. The node n_5 has n_4 in its label, so we introduce the arc a_{45}, and so forth. The resulting arborescence is shown in Figure 5.17.

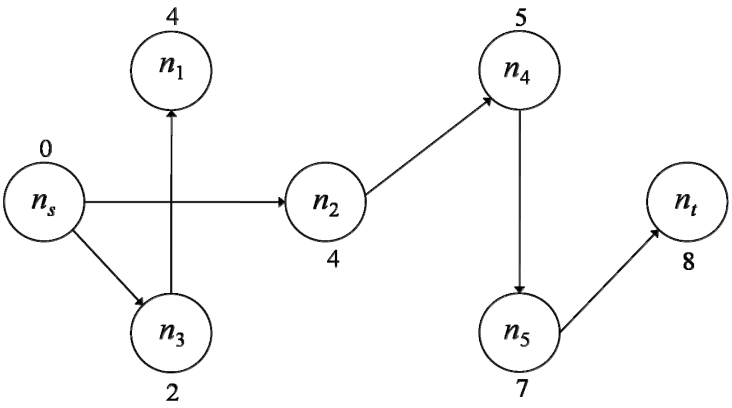


Figure 5.17

The numbers next to the nodes are the lengths of the shortest paths from the source n_s to all nodes in the network.

Sometimes (as for instance in location models), it is required to determine the paths between all pairs of nodes. Clearly, Dijkstra's technique could be used by considering one node as a source, determine the arborescence of all shortest paths rooted at that node, and then repeat the process with all nodes as roots. This is somewhat tedious, and there is a very efficient technique, called the *Floyd-Warshall method*, that performs this task directly. All it requires are some matrix operations. The method starts and works with the direct distance matrix $\mathbf{C}^0$, which includes all node-to-node distances of the original problem. An iterative step in iteration k can then be described as follows. In the first iteration, we use row and column 1 as the *key row* and *column*. We then compare the shortest presently known distance between node n_i and node n_j with a detour that uses node n_1. The shorter of the two distances is then used as the new shortest known distance. This process is repeated for all pairs of nodes. That way, we reach the revised distance matrix $\mathbf{C}^1$. We now use the second row and column as key row and column, and compare all distances with the detour via n_2. In the matrix, this is simply done by comparing each element with the sum of its corresponding element in the key row and key column. This process is repeated until all nodes have been used as detour node once. The resulting matrix $\mathbf{C}^n$ (given that there graph has n nodes) is then the matrix of shortest paths.

We will illustrate this procedure by means of an example. The graph and its distances of the example are shown in Figure 5.18.

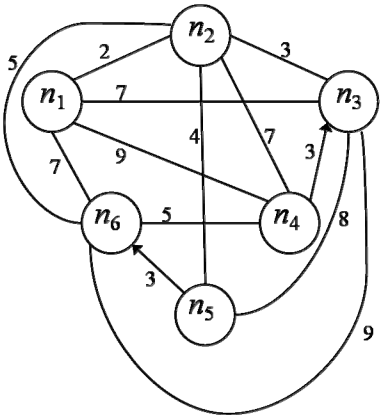


Figure 5.18

The direct distance matrix of the mixed graph in the example is

$$\mathbf{C}^0 = \begin{bmatrix} 0 & 2 & 7 & 9 & \infty & 7 \\ 2 & 0 & 3 & 7 & 4 & 5 \\ 7 & 3 & 0 & \infty & 8 & 9 \\ 9 & 7 & 3 & 0 & \infty & 5 \\ \infty & 4 & 8 & \infty & 0 & 3 \\ 7 & 5 & 9 & 5 & \infty & 0 \end{bmatrix}$$

Note that in case of undirected graphs, the direct distance matrix is symmetric. In the first iteration $k = 1$, we will consider the node n_1 as a possible detour node. This is done by using the first row and column as the key row and column. This row and column is copied without a change into the next matrix. For all other elements, we compare the present distance with the one that uses n_1 as a detour node. As an example, the lowest known distance from n_2 to n_3 that is presently known is $c_{23} = 3$. This is compared with the detour that uses node 1, i.e., the path that uses the arcs (n_2, n_1) and (n_1, n_3), whose lengths are 2 and 7, respectively. In other words, we now compare 3 with $2 + 7 = 9$. Since the original length is shorter, we keep it and continue with the next pair of nodes. Next, we compare the distances from n_2 to n_4, without and with detour via n_1. We find that $c_{24} = 7 < 2 + 9 = c_{21} + c_{14}$, so that again, we make no change. We continue this way and find no changes, until we reach the connection from n_3 to n_4. At present, $c_{34} = \infty$, i.e., there is no direct connection. We compare this distance with the detour that uses n_1, i.e., the arcs (n_3, n_1) and (n_1, n_4). Those distances are 7 and 9, so that we are now able to reach n_4 from n_3 on a path of length $7 + 9 = 16$. This turns out to be the only change from matrix $\mathbf{C}^0$ to matrix $\mathbf{C}^1$, which is indicated in $\mathbf{C}^1$ by an asterisk.

$$\mathbf{C}^1 = \begin{bmatrix} 0 & 2 & 7 & 9 & \infty & 7 \\ 2 & 0 & 3 & 7 & 4 & 5 \\ 7 & 3 & 0 & 16* & 8 & 9 \\ 9 & 7 & 3 & 0 & \infty & 5 \\ \infty & 4 & 8 & \infty & 0 & 3 \\ 7 & 5 & 9 & 5 & \infty & 0 \end{bmatrix}$$

Starting now with $\mathbf{C}^1$, the process is repeated by using the second row and column as key row(column) and comparing all known distances in $\mathbf{C}^1$ with the detour that uses n_2 as detour node. Again, we copy the key row and column without any changes to C^2. Here are some of the computations. For the connection from n_1 to n_3, we presently have a length of 7, which we compare with the detour via node n_2, which has a length of 2 + 3 = 5, so that the new distance is shorter. Similarly, the distance from n_1 to n_4 without the detour via n_2 is 9, with the detour it is 2 + 7 = 9, a tie. The distance from n_1 to n_5 without the detour is ∞, with the detour, it is 2 + 4 = 6. The results are shown in the matrix $\mathbf{C}^2$, again with all changes indicated by an asterisk.

$$\mathbf{C}^2 = \begin{bmatrix} 0 & 2 & 5* & 9 & 6* & 7 \\ 2 & 0 & 3 & 7 & 4 & 5 \\ 5* & 3 & 0 & 10* & 7* & 8* \\ 9 & 7 & 3 & 0 & 11* & 5 \\ 6* & 4 & 7* & 11* & 0 & 3 \\ 7 & 5 & 8* & 5 & 9* & 0 \end{bmatrix}$$

The remaining iterations are shown without further comment.

$$\mathbf{C}^3 = \begin{bmatrix} 0 & 2 & 5 & 9 & 6 & 7 \\ 2 & 0 & 3 & 7 & 4 & 5 \\ 5 & 3 & 0 & 10 & 7 & 8 \\ 8* & 6* & 3 & 0 & 10* & 5 \\ 6 & 4 & 7 & 11 & 0 & 3 \\ 7 & 5 & 8 & 5 & 9 & 0 \end{bmatrix}, \quad \mathbf{C}^4 = \begin{bmatrix} 0 & 2 & 5 & 9 & 6 & 7 \\ 2 & 0 & 3 & 7 & 4 & 5 \\ 5 & 3 & 0 & 10 & 7 & 8 \\ 8 & 6 & 3 & 0 & 10 & 5 \\ 6 & 4 & 7 & 11 & 0 & 3 \\ 7 & 5 & 8 & 5 & 9 & 0 \end{bmatrix},$$

$$\mathbf{C}^5 = \begin{bmatrix} 0 & 2 & 5 & 9 & 6 & 7 \\ 2 & 0 & 3 & 7 & 4 & 5 \\ 5 & 3 & 0 & 10 & 7 & 8 \\ 8 & 6 & 3 & 0 & 10 & 5 \\ 6 & 4 & 7 & 11 & 0 & 3 \\ 7 & 5 & 8 & 5 & 9 & 0 \end{bmatrix}, \text{ and } \mathbf{C}^6 = \begin{bmatrix} 0 & 2 & 5 & 9 & 6 & 7 \\ 2 & 0 & 3 & 7 & 4 & 5 \\ 5 & 3 & 0 & 10 & 7 & 8 \\ 8 & 6 & 3 & 0 & 8* & 5 \\ 6 & 4 & 7 & 8* & 0 & 3 \\ 7 & 5 & 8 & 5 & 9 & 0 \end{bmatrix}$$

The matrix $\mathbf{C}^6$ now includes the lengths of the shortest paths between all pairs of nodes. We should note that it is also possible to construct a second set of matrices parallel to the computations made above, so as to keep track not only of the lengths of the shortest paths, but also the paths themselves. This is, however, beyond the scope of this volume. Interested readers are referred to books, such as Eiselt and Sandblom (2000).

5.4 Spanning Tree Problems

Similar to the models in the previous sections in this chapter, the problem behind spanning trees is easily explained. Suppose there is an undirected graph that includes the potential edges that connect the nodes of the graph. The values associated with the edges indicate their costs. The problem is now to choose some of the edges and include them in the solution, so that the costs are minimized, while the network remains connected. This may result in a graph, in which the path from one node to another leads through many intermediate nodes.

It is apparent that what we are looking for is a tree, as any graph with less edges than a tree has will no longer be connected, while any graph that has one or more edges than a tree will include at least one cycle, which is unnecessary.

Problems of this nature occur whenever it is very expensive to establish edges. Typical examples are road networks, networks of power lines, or networks of sewer tunnels. The example shown in Figure 5.19 shows all possible connections that may be established, coupled with their respective costs.

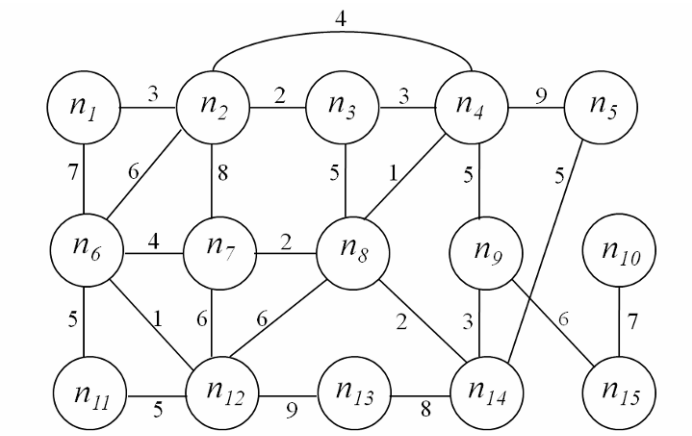


Figure 5.19

The task is now to find a subset of existing edges (the edges that belong to connections that will actually be built), so as to minimize the total amount of money necessary to connect all nodes with each other. The resulting connected subgraph of the original graph with n nodes is a *minimal spanning tree* of the given graph. Given that the result will be a tree, one can show that the optimal solution will include exactly $(n-1)$edges.

A variety of solution methods exist for this type of problem. Here, we describe the *Kruskal technique*, which was first published in 1956. Actually, the method is nothing but a Greedy technique. However, while the Greedy heuristic typically finds only approximate solutions, it is guaranteed to find an optimal solution for this problem. It can be described as follows. We first sort the edges in order of nondecreasing arc values, where ties are broken arbitrarily. Starting with the edge that has the lowest arc value, we introduce one arc at a time, provided it does not form a cycle with the already existing edges. The procedure continues until $(n-1)$ arcs are included in the solution.

In the example of Figure 5.19, we order the edges, which results in Table 5.2. The table shows the edges and their costs in nondecreasing order.

Table 5.2: Edges of Figure 5.19 in order of their value

Arc	a_{48}	$a_{6,12}$	a_{23}	a_{78}	$a_{8,14}$	a_{12}	a_{34}	$a_{6,11}$	$a_{9,14}$	a_{24}	a_{67}
Cost	1	1	2	2	2	3	3	3	3	4	4
Inserted?	✓	✓	✓	✓	✓	✓	✓	✓	✓	No	✓

Arc	a_{38}	a_{49}	$a_{5,14}$	$a_{11,12}$	a_{26}	$a_{7,12}$	$a_{8,12}$	$a_{9,15}$	a_{16}	$a_{10,15}$	a_{27}
Cost	5	5	5	5	6	6	6	6	7	7	8
Inserted?	No	No	✓	No	No	No	No	✓	No	✓	No

Table 5.2 (continued)

Arc	$a_{13,14}$	a_{45}	$a_{12,13}$
Cost	8	9	9
Inserted?	✓	No	No

Table 5.2 indicates which edges are introduced and which are not. In order to visualize the process, consider Figure 5.20. The solid lines are edges that are introduced in the first nine steps, the broken-and-dotted lines are edges that are introduced in the next two steps, and the dotted lines are edges introduced after that. Note that after edge $a_{13,14}$ is introduced, we have introduced 14 edges, and the process terminates.

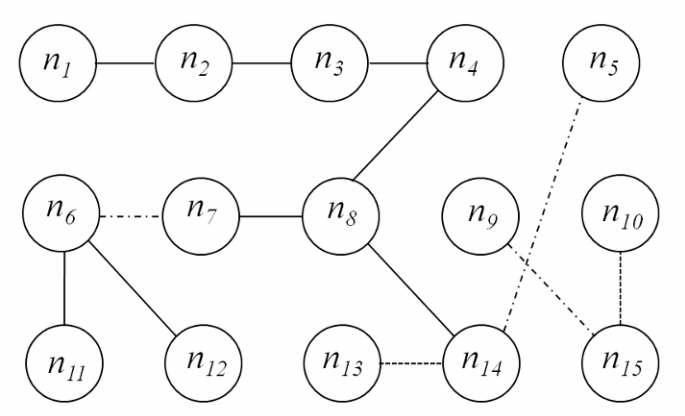


Figure 5.20

5.5 Routing Problems

Routing problems are among the most frequently used network models in practice. We distinguish between two classes of routing models: *arc routing* and *node routing* models. The first example of arc routing was provided by Euler and his "Königsberg Bridge Problem" described at the beginning of this chapter. The idea common to all arc routing problems is to find a tour in a given graph, so that each arc is used on the tour exactly or at least once. Similarly, in node routing problems, the idea is to find a tour that starts at some node and returns to it, while using each node exactly or at least once in the process. Combinations of the two classes are vehicle routing problems, which belong to the most difficult routing problems.

The best-known arc routing problem is the *Chinese Postman Problem*. The name is due to Meigu Guan, who in the course of the "cultural" revolution in China was assigned to the position of a postal worker in the early 1960s. There, he considered the problem of a letter carrier, who would pick up the mail at some point (a node in the street network), and deliver it to the individual households by walking along each street at least once. The objective of the model is to minimize total distance, and the constraints ensure that mail is delivered to the houses on all streets of the network. While some version of the model are easy to solve, others remain difficult. There are many important and popular applications of Chinese Postman Problems, including street cleaning and snow removal. Clearly, those problems are more difficult, as they will include hierarchies of streets, e.g., highways are typically plowed after a snowstorm before small neighborhood streets.

Similar to arc routing, the field of node routing has a long history. It starts with Hamilton's "trip around the world" developed in 1856, a game in which players have to find a tour that visits all desirable places (the nodes in a graph) at least once. The best-known version of node routing is the famed *traveling salesman problem*, surely one of the most popular models in all of operations research. The story (not a real application, but a scenario that give the model its name) is that a traveling salesman attempts to sell his goods in a number of cities in his region. He must visit each city once and must return to the place he started from. In order to have as much time as possible with the customers, the objective is to minimize the total time (or distance) of the tour. Applications of traveling salesman problems abound, many of them seemingly unrelated. One such example is the drilling of holes into sheet metal with the use of an automated drill press. Drilling the hole takes the same amount of time regardless of the sequence, in which the holes are drilled, so that the objective is to minimize the amount of time it takes to move the metal into the position, in which the next hole is to be drilled. This is nothing but a traveling salesman problem.

Due to space limitations, we will only deal with traveling salesman problems in this section. Good algorithms for some versions of the Chinese postman problem exist, while the traveling salesman problem is notoriously difficult to solve.

At first glance, formulating a traveling salesman problem appears easy. We need to formulate zero-one variables x_{ij} that assume a value of one, if the arc a_{ij} is part of the tour, and 0 otherwise. The objective function is then simply to minimize the sum of arc values, each multiplied with their respective binary variable. As far as constraints go, we have to ensure that the traveling salesman tour enters each node exactly once and that it leaves each node exactly once. In order to explain the formulation, consider as an example the graph in Figure 5.21.

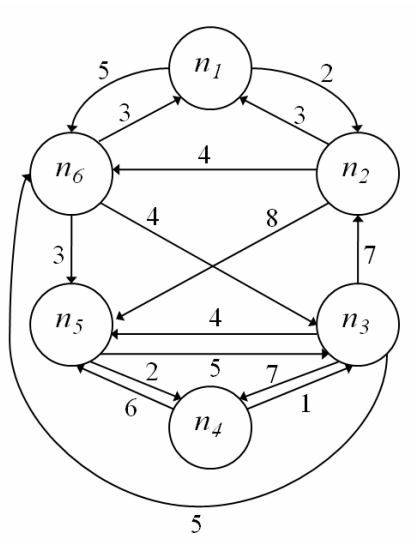


Figure 5.21

The formulation as described above is then as follows:

Min $z = 2x_{12} + 5x_{16} + 3x_{21} + 8x_{25} + 4x_{26} + 7x_{32} + 7x_{34} + 5x_{36} + 4x_{35} + 1x_{43} + 6x_{45} + 5x_{53} + 2x_{54} + 3x_{61} + 4x_{63} + 3x_{65}$

s.t. $x_{12} + x_{16} = 1$

$x_{21} + x_{25} + x_{26} = 1$

$x_{32} + x_{34} + x_{35} + x_{36} = 1$

$x_{43} + x_{45} = 1$

$x_{53} + x_{54} = 1$

$x_{61} + x_{63} + x_{65} = 1$

$x_{21} + x_{61} = 1$

$x_{12} + x_{32} = 1$

$x_{43} + x_{53} + x_{63} = 1$

$x_{34} + x_{54} = 1$

$x_{25} + x_{35} + x_{45} + x_{65} = 1$

$x_{16} + x_{26} + x_{36} = 1$

$x_{ij} = 0$ or 1 for all i, j.

The first six constraints in the above formulation force the outflow of each node to be equal to one (meaning that the traveling salesman tour will leave each node exactly once), while the second set of six constraints requires that the inflow into each node equals one (meaning that the tour enters each node exactly once). If the original graph would have direct connections between all pairs of nodes, the above formulation would actually be identical to that of an assignment problem. We know that for this type of problem, the zero-one constraints are satisfied without us requiring them, so that the problem can be solved as a standard linear

programming problem. The result we find includes the following nonzero variables: $\bar{x}_{12} = \bar{x}_{21} = \bar{x}_{36} = \bar{x}_{43} = \bar{x}_{54} = \bar{x}_{65} = 1$, and all other variables equal zero. The value of the objective function for this solution equals $\bar{z} = 16$.

It is easily apparent that this solution does satisfy the constraints, but is not what we were looking for: rather than one tour, the solution includes two subtours ($n_1 - n_2 - n_1$) and ($n_3 - n_6 - n_5 - n_4 - n_3$). What we will have to add are so-called *subtour elimination constraints*. There are different ways of doing this, but the most efficient sets of such constraints require a huge number of constraints. To be precise, for a graph of n nodes, there are 2^n such subtour elimination constraints. As a result, the modeler will refrain from including all of these constraints from the beginning, but rather solve the problem without them. Then, if the solution has no subtours, we are done. Otherwise, a single relevant subtour elimination constraint is introduced. This is the process we follow here.

The idea is now this. First, we define N_s as the set of nodes in the chosen subtour, and let $\overline{N}_s$ denote the complement of this set. Note that the present solution has no arc leading out of N_s to any node in $\overline{N}_s$. Therefore, we define a constraint that requires at least one arc in the solution to lead out of a node in N_s to a node in $\overline{N}_s$. In our example, choose the subtour ($n_1 - n_2 - n_1$), so that $N_s = \{n_1, n_2\}$ and $\overline{N}_s = \{n_3, n_4, n_5, n_6\}$. The set $\{a_{16}, a_{25}, a_{26}\}$ includes all arcs that lead from N_s to $\overline{N}_s$, so that we can formulate the constraint

$$x_{16} + x_{25} + x_{26} \geq 1.$$

We now solve the problem again with this additional constraint. Note that now due to the additional constraint the "assignment structure" of the problem is lost and it is necessary to include the zero-one requirements for all variables, which makes the problem considerably more difficult. The optimal solution of the problem is $\bar{x}_{12} = \bar{x}_{26} = \bar{x}_{35} = \bar{x}_{43} = \bar{x}_{54} = \bar{x}_{61} = 1$ with an objective value of $\bar{z} = 16$ (so apparently, there were alternative optimal solutions to the problem in the first step). This means that our tour is ($n_1 - n_2 - n_6 - n_1$) and ($n_3 - n_5 - n_4 - n_3$), meaning that we have successfully eliminated the previous subtour, but now have a solution with another subtour, so that another subtour elimination constraint must be added.

Given the subtour ($n_1 - n_2 - n_6 - n_1$), our sets are $N_s = \{n_1, n_2, n_6\}$ and $\overline{N}_s = \{n_3, n_4, n_5\}$, so that the set of arcs from N_s to $\overline{N}_s$ is $\{a_{25}, a_{63}, a_{65}\}$. The additional constraint can then be written as

$$x_{25} + x_{63} + x_{65} \geq 1.$$

Solving the problem again results in the solution $\bar{x}_{16} = \bar{x}_{21} = \bar{x}_{32} = \bar{x}_{43} = \bar{x}_{54} = \bar{x}_{65} = 1$ with a value of the objective function $\bar{z} = 21$. This solution includes the tour ($n_1 - n_6 - n_5 - n_4 - n_3 - n_2 - n_1$), which no longer includes a subtour. Thus this is the optimal solution.

While this procedure may be feasible for small and medium-sized problems, it is not for large-scale applications. Here, we may resort to heuristic algorithms. In the simplest case, we may use the Greedy algorithm to find a tour. In this application, we would start the Greedy algorithm with some node, find the nearest neighbor (provided it does not result in a subtour), move on to the next neighbor, again avoiding, subtours, and so forth. Note that the number of degrees of freedom is constantly decreasing while we make choices. (Not that there is anything new in that: Whenever you make a choice such as spending money on some item, you will have less choices, i.e., money, for future decisions).

In order to explain the procedure, consider the distance matrix

$$\mathbf{D} = \begin{bmatrix} 0 & 4 & 7 & 3 & 2 & 6 \\ 3 & 0 & 4 & 9 & 6 & 8 \\ 5 & 4 & 0 & 8 & 3 & 4 \\ 6 & 8 & 3 & 0 & 2 & 5 \\ 1 & 7 & 2 & 3 & 0 & 4 \\ 8 & 7 & 4 & 9 & 4 & 0 \end{bmatrix}$$

Arbitrarily starting with the node n_1, the nearest neighbor, i.e., the smallest element in the first row of **D** (other than the element $d_{11} = 0$, which would just lead from n_1 back to itself), is the connection to n_5 at a distance of 2. From n_5, i.e., in row 5 of the matrix, the nearest neighbor is n_1 at a distance of 1. However, this connection would create a subtour, so we look for the next-shortest distance. It is the link from n_5 to n_3 with a distance of 2. From n_3, the nearest neighbor is n_5 with a distance of 3, but going back to n_5 would create a subtour. The next shortest distances are those to n_2 and n_6, both with a distance of 4. Any tie-breaking rule can be used, here, we choose n_2. From n_2, the nearest neighbor is n_1, which cannot be chosen, as it would create a subtour. The next-nearest neighbor is n_3, which cannot be chosen for the same reason. The next-nearest neighbor is n_5, which is also not eligible. The next-nearest neighbor is n_6 at a distance of 8, which must be chosen. Note how the lack of degrees of freedom forces us to choose undesirable links at the later stages of the algorithm. Actually, at this point there is no degree of freedom left, and we have to continue on to n_4, the last remaining node, and from there return to n_1. The two distances are 9 and 6, respectively, so that the entire tour is of length $\ell = 31$. In summary, the tour is $n_1 - n_5 - n_3 - n_2 - n_6 - n_4 - n_1$.

In a "Phase 2" procedure, we can now attempt to improve the solution found earlier. One way to do this is a "pairwise exchange" or "swap" method. It very simply exchanges two (usually, but not necessarily, adjacent) nodes on the tour. In the above tour, we may try to avoid the long distance from n_6 to n_4 by switching the order, in which these two nodes are visited on the tour. Doing so results in the tour $n_1 - n_5 - n_3 - n_2 - n_4 - n_6 - n_1$, which has length $\ell = 30$, which is better, so that this new tour becomes the starting point for further improvements.

Starting with the new tour, we may now attempt to avoid the direct connection from n_2 to n_4, which is also a very long leg of the tour. Switching the order of these two nodes results in the tour $n_1 - n_5 - n_3 - n_4 - n_2 - n_6 - n_1$, which is of length $\ell = 36$. This is higher than the previous (best known) solution, so that the switch is not made. This swap process can be continued, until no swap change is able to further decrease the length of the tour.

It is sometimes useful to use what is known as a *multistart procedure*. This means that rather than starting with some random node (n_1 in the above example), finding a solution and then trying to improve it, we may try out all different nodes as potential starting points. With each such node, we obtain a tour. We would then choose the best tour, and try to find improvements from there.

In the above example, we could use the Greedy algorithm to find tours starting with each of the six nodes. In addition to the tour that starts with n_1, which has already been determined above, we obtain the tours $n_2 - n_1 - n_5 - n_3 - n_6 - n_4 - n_2$ of length $\ell = 28$, $n_3 - n_5 - n_1 - n_4 - n_6 - n_2 - n_3$ of length $\ell = 23$, $n_4 - n_5 - n_1 - n_2 - n_3 - n_6 - n_4$ of length $\ell = 24$, $n_5 - n_1 - n_4 - n_3 - n_2 - n_6 - n_5$ of length $\ell = 23$, and $n_6 - n_3 - n_5 - n_1 - n_4 - n_2 - n_6$ of length $\ell = 27$. The tours that start with n_3 and n_5 are best, and the swap process would start with them. Any of these heuristic procedures is computationally cheap and, once some tour has been obtained, the process can be terminated at any point in time.

Exercises

Problem 1 (maximal flow algorithm, minimal cut): Consider the network in Figure 5.22, in which the numbers next to the directed arcs denote the capacity of the arcs.

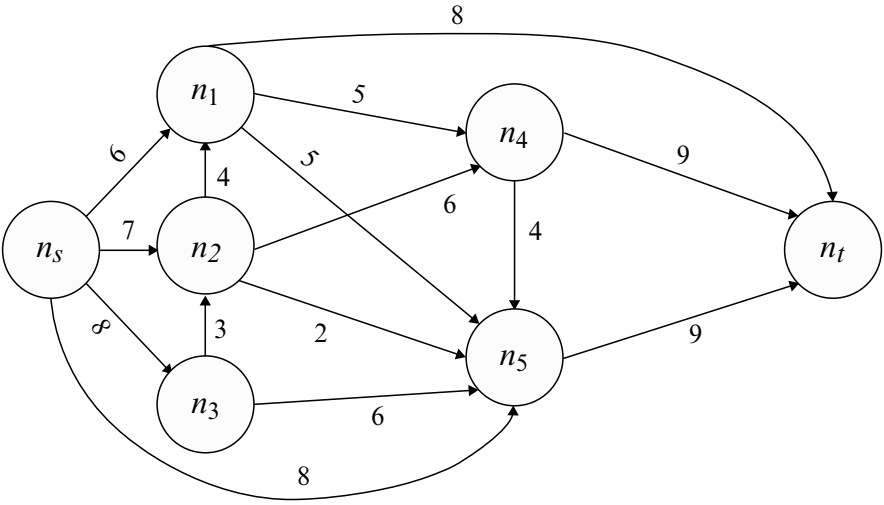


Figure 5.22

Use the method by Ford and Fulkerson to determine a maximal flow. Show the flow pattern. What is the value of the maximal flow? Which arcs are in the minimal cut you have found?

Solution: Starting with a flow of zero at all arcs, we first increase the flow on the following paths:

path (n_s, n_1, n_t): 6 units,
path (n_s, n_2, n_4, n_t): 6 units,
path (n_s, n_3, n_5, n_t): 6 units, and
path (n_s, n_5, n_t): 3 units.

The flow pattern is then shown in the network in Figure 5.23, where the two numbers next to the arcs indicate the arc's capacity and its present flow, respectively.

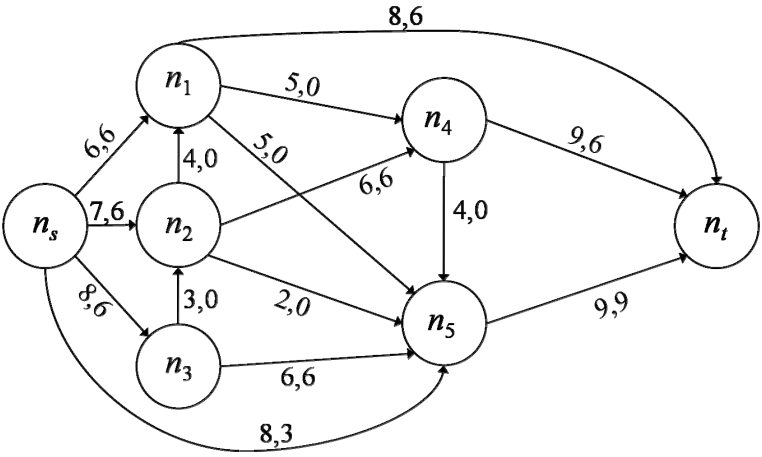


Figure 5.23

The next step consists of an increase the flow on the following two paths:

path (n_s, n_3, n_2, n_1, n_t): 2 units, and

path (n_s, n_2, n_1, n_4, n_t): 1 unit.

The resulting flow pattern is shown in Figure 5.24.

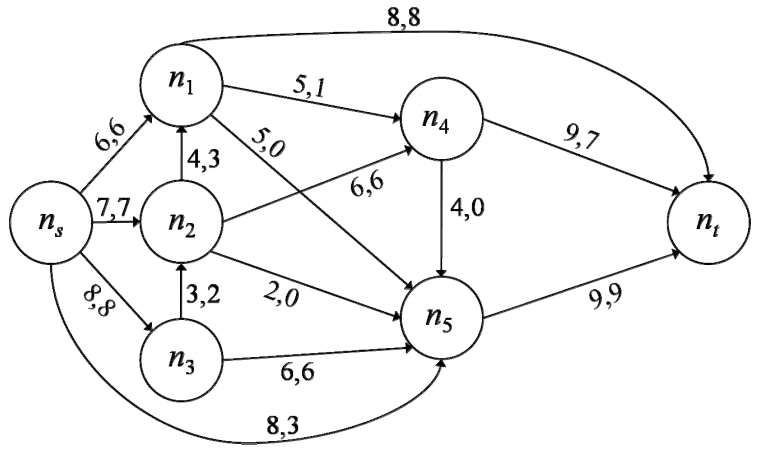


Figure 5.24

At this point, there are no more degrees of freedom, and labeling can only be done on the path (n_s, n_5, n_3 (backward labeling at this point), n_2, n_1, n_4, n_t). The flow on this path can be changed by one unit, resulting in the flow pattern shown in the network in Figure 5.25.

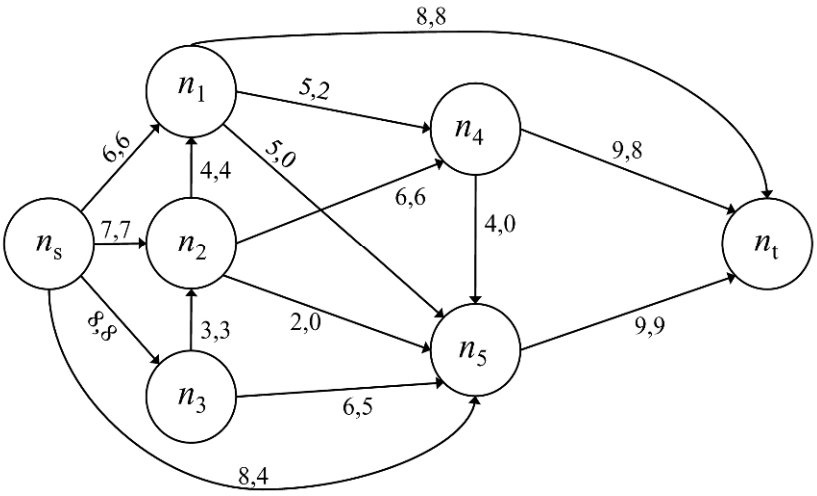


Figure 5.25

Starting with the pattern in Figure 5.25, we can label the nodes n_s, n_3, and n_5. At this point, a nonbreakthrough occurs, and we can conclude that the flow pattern in Figure 5.25 is maximal. The corresponding flow value is 25. The minimal cut include the capacities of all arcs that lead from labeled to unlabeled nodes. Here the minimal cut includes the arcs (n_s, n_1), (n_s, n_2), (n_3, n_2), and (n_5, n_t). Note that the Ford and Fulkerson method only finds the cut that is closest to the source. In this example, another minimal cut exists with arcs (n_s, n_1), (n_2, n_1), (n_2, n_4), and (n_5, n_t); yet another cut is (n_1, n_t), (n_4, n_t), (n_5, n_t).

Problem 2 (formulation of a feasible flow problem with node constraints): Consider the network shown in Figure 5.26, where the numbers next to the arcs consists of the lower and upper bound on the flow in the arc, while the double-digit numbers next to the arcs indicate the per-unit cost of the flow through the arc.

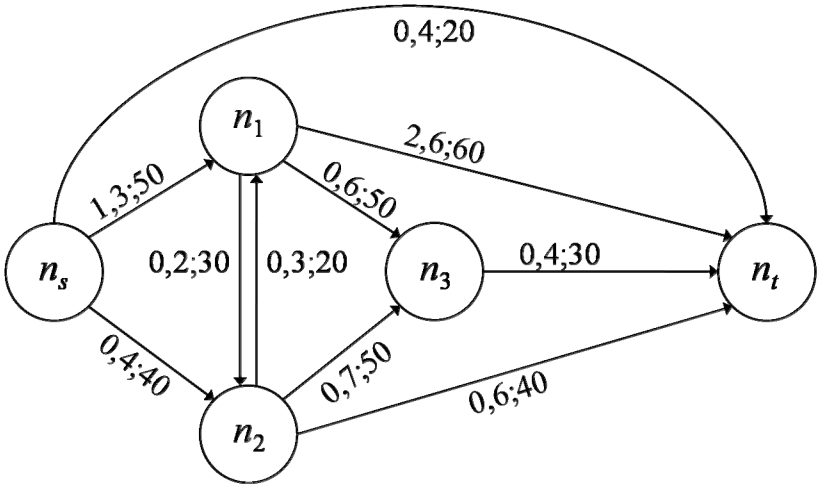


Figure 5.26

In addition, we want to ensure that a total of 5 units flow from the source to the sink and exactly two units flow through the node n_3. Formulate.

Solution: Defining variables x_{ij} as the flow from node i to node j on the arc connecting the two nodes, we can formulate the problem as follows.

P: Min $z = 20x_{st} + 50x_{s1} + 40x_{s2} + 20x_{12} + 50x_{13} + 60x_{1t} + 20x_{21} + 50x_{23} + 40x_{2t} + 30x_{3t}$

s.t. $x_{st} + x_{s1} + x_{s2} = 5$ (forcing 5 units through the network)
$x_{s1} + x_{21} - x_{12} - x_{13} - x_{1t} = 0$ (conservation equation for node n_1)
$x_{s2} + x_{12} - x_{21} - x_{23} - x_{2t} = 0$ (conservation equation for node n_2)
$x_{13} + x_{23} - x_{3t} = 0$ (conservation equation for node n_3)
$x_{13} + x_{23} = 2$ (or, equivalently, $x_{3t} = 2$: forces a flow of 2 through node n_3)

$$\left.\begin{array}{l} x_{st} \le 4 \\ x_{s1} \le 3 \\ x_{s2} \le 4 \\ x_{12} \le 2 \\ x_{13} \le 6 \\ x_{1t} \le 6 \\ x_{21} \le 3 \\ x_{23} \le 7 \\ x_{2t} \le 6 \\ x_{3t} \le 4 \end{array}\right\} \quad \text{(upper limits on arc flows)}$$

$$\left.\begin{array}{l} x_{s1} \geq 1 \\ x_{1t} \geq 2 \end{array}\right\} \text{(lower limits on arc flows)}$$

$x_{ij} \geq 0$ for all variables.

The optimal flow pattern is shown in Figure 5.27, where the numbers next to the arcs are the arc flows. The total cost for the flow pattern are \$480.

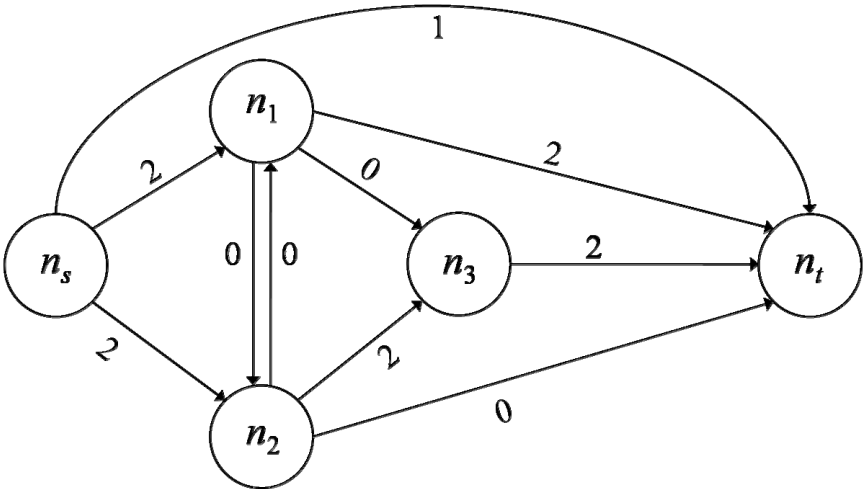


Figure 5.27

Changing the requirement regarding the throughput of node n_3 from "exactly 2 units" to "at most 2 units" results in a flow of 3 units on the arc from n_s to n_t, and 2 units on the path from n_s to n_1 and n_t. The total costs of this solution are \$280.

Problem 3 (shortest path, Dijkstra method): Consider the network in Figure 5.28 and determine all shortest paths from n_s to all other nodes.

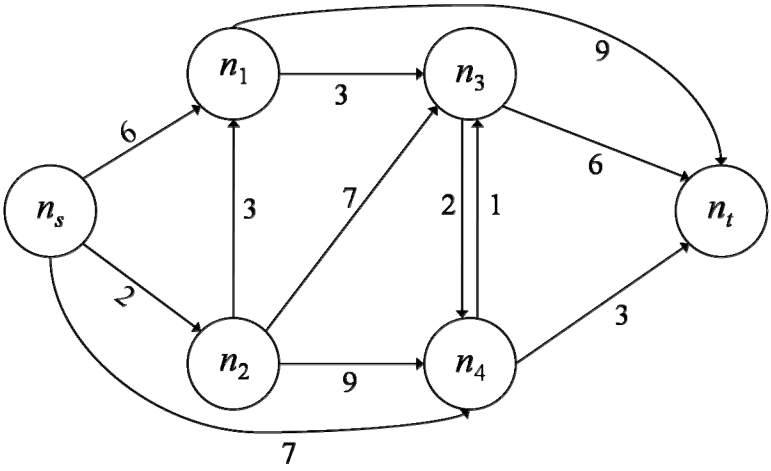


Figure 5.28

Solution: Table 5.3 shows the labeling process.

Table 5.3: Labels of the nodes in the shortest path example

Step #	$L(n_s)$	$L(n_1)$	$L(n_2)$	$L(n_3)$	$L(n_4)$	$L(n_t)$
0	$(n_s, 0)$*	(n_1, ∞)	(n_2, ∞)	(n_3, ∞)	(n_4, ∞)	(n_t, ∞)
1		$(n_s, 6)$	$(n_s, 2)$*	(n_3, ∞)	(n_4, ∞)	(n_t, ∞)
2		$(n_2, 5)$*		$(n_2, 9)$	$(n_2, 11)$	(n_t, ∞)
3				$(n_1, 8)$*	$(n_2, 11)$	$(n_1, 14)$
4					$(n_3, 10)$*	$(n_1, 14)$, or $(n_3, 14)$
5						$(n_4, 13)$

The tree with the shortest distances is shown in Figure 5.29. Note that in this example, the tree is a simple path.

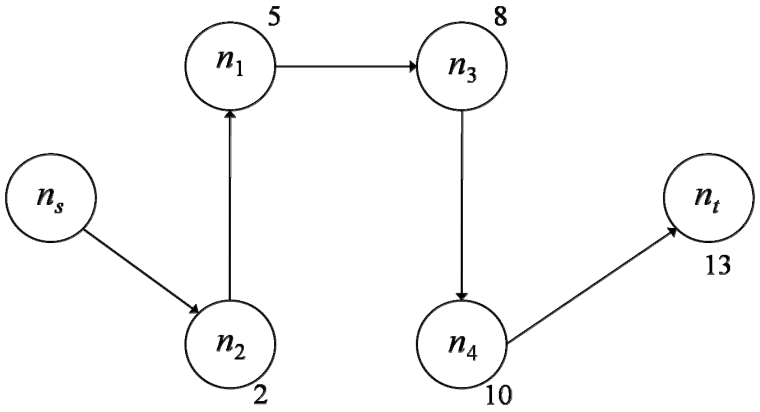


Figure 5.29

Problem 4 (shortest path, Floyd-Warshall method): Consider the network shown in Figure 5.30.

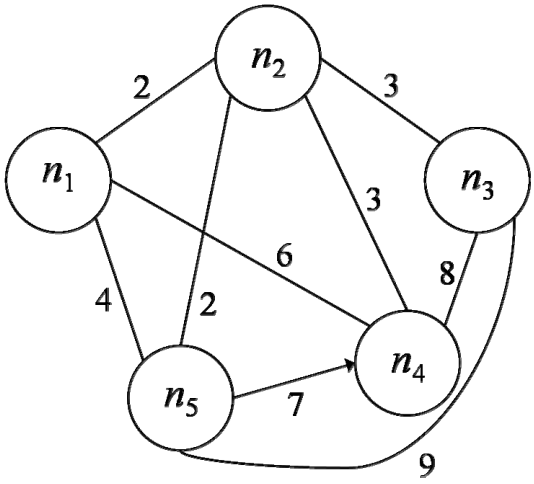


Figure 5.30

Use the Floyd-Warshall algorithm to determine the shortest paths between all pairs of nodes.

Solution: The direct distance matrix is

$$\begin{bmatrix} 0 & 2 & \infty & 6 & 4 \\ 2 & 0 & 3 & 3 & 2 \\ \infty & 3 & 0 & 8 & 9 \\ 6 & 3 & 8 & 0 & \infty \\ 4 & 2 & 9 & 7 & 0 \end{bmatrix}$$

The first two steps result in the following approximations (where the elements with a "*" indicate changes and elements with a "=" indicate a tie in the step that led to the new solution):

$$\begin{bmatrix} \boxed{0} & 2 & \infty & 6 & 4 \\ 2 & 0 & 3 & 3 & 2 \\ \infty & 3 & 0 & 8 & 9 \\ 6 & 3 & 8 & 0 & \infty \\ 4 & 2 & 9 & 7 & 0 \end{bmatrix} \quad \begin{bmatrix} 0 & \boxed{2} & \infty & 6 & 4 \\ \boxed{2} & 0 & 3 & 3 & 2 \\ \infty & 3 & 0 & 8 & 9 \\ 6 & 3 & 8 & 0 & 10* \\ 4 & \boxed{2} & 9 & 7 & 0 \end{bmatrix} \quad \begin{bmatrix} 0 & 2 & \boxed{5*} & 5* & 4 \\ 2 & 0 & 3 & 3 & 2 \\ \boxed{5*} & 3 & 0 & \boxed{6*} & \boxed{5*} \\ 5* & 3 & \boxed{6*} & 0 & 5* \\ 4 & 2 & \boxed{5*} & 5* & 0 \end{bmatrix}$$

After this, there are no further changes, so that the last matrix is indeed the shortest path matrix.

Problem 5 (minimal spanning tree): A manufacturing firm wants to connect its nine plants by rail. The cost of establishing the line segments are shown in the table below, where a number in row i and column j indicates the fixed costs to connect plant i with plant j.

$$\begin{bmatrix} 0 & 23 & 64 & 15 & 39 & 66 & 18 & 84 & 93 \\ 23 & 0 & 35 & 27 & 56 & 63 & 29 & 91 & 37 \\ 64 & 35 & 0 & 96 & 62 & 45 & 33 & 72 & 48 \\ 15 & 27 & 96 & 0 & 38 & 22 & 74 & 61 & 88 \\ 39 & 56 & 62 & 38 & 0 & 42 & 95 & 54 & 67 \\ 66 & 63 & 45 & 22 & 42 & 0 & 25 & 43 & 47 \\ 18 & 29 & 33 & 74 & 95 & 25 & 0 & 51 & 32 \\ 84 & 91 & 72 & 61 & 54 & 43 & 51 & 0 & 26 \\ 93 & 37 & 48 & 88 & 67 & 47 & 32 & 26 & 0 \end{bmatrix}$$

Solution: Putting the costs in nondecreasing order (ties are broken arbitrarily) results in the sequence 15, 18, 22, 23, 25, 26, 27, 29, 32, 33, 35, 37, 38, 39, 42, 43, 45, 47, 48, 51, 54, 56, 61, 62, 63, 64, 66, 67, 72, 74, 84, 88, 91, 93, 95, and 96.

Starting at the beginning, we introduce connections (1, 4), (1, 7), (4, 6), and (1, 2) with costs of 15, 18, 22, and 23. The next cheapest connection is (6, 7) with costs of 25. However, plants 6 and 7 are already connected. the next connection is (8, 9) with costs of 26, which we introduce. The next connection on the list is (2, 4) with costs of 27. Plants 2 and 4 are already connected, so that we reject this connection and continue. The next connection is (2, 7) with costs of 29, which is also rejected. The next connection is (7, 9) with costs of 32, which is introduced. This is followed by connection (3, 7) with costs of 33, which is introduced. This is followed by the connections (2, 3) with costs of 35 and (2, 9) with costs of 37; both of which are rejected. The connection (4, 5) with costs 38 connects two previously unconnected nodes, and thus it is introduced. At this point, eight arcs have been introduced and all plants are connected. Thus, an optimal solution has been determined. Its costs are 207.

Problem 6 (heuristics applied to a traveling salesman problem): A pharmaceutical company uses glass containers to store their chemicals. Over time, they reuse the containers, but if they do, they are required to clean them. The cleaning costs depend on what was stored in the container before, and what will be stored next. The cleaning costs of the six chemicals *A*, *B*, *C*, *D*, *E*, and *F* are shown in the following matrix:

$$\begin{bmatrix} 0 & 6 & 5 & 3 & 5 & 4 \\ 2 & 0 & 7 & 9 & 3 & 8 \\ 5 & 5 & 1 & 6 & 4 & 2 \\ 4 & 2 & 3 & 2 & 1 & 3 \\ 9 & 6 & 4 & 3 & 0 & 7 \\ 2 & 3 & 8 & 1 & 6 & 0 \end{bmatrix}$$

For example, if chemical *C* is stored in a glass that was used for *F* before, then the cleaning costs are 8 (from *F* to *C*, i.e., element (*F*, *C*), in the 6th row and 3rd column.).

(a) Assume that presently, chemical *D* is stored in a container. Use the Greedy algorithm to determine a sequence that has each of the six chemicals stored in a container exactly once. (Ties are broken arbitrarily). Clearly specify the sequence that results from the application of the Greedy algorithm. What are the costs associated with the sequence?
(b) Use the pairwise exchange method to improve the solution determined under (a). Examine all pairs until an exchange results in an improvement. Make this improvement and stop (even though additional improvements may be possible).

Solution:

(a) Starting with D, we obtain the sequence $D - E - C - F - A - B - D$. The length of the tour (here: the cost) is $\ell = 1 + 4 + 2 + 2 + 6 + 9 = 24$.

(b) In this application, the starting point at D is fixed. The following swap steps can be made:

Swap E and C. Result: $D - C - E - F - A - B - D$, length $\ell = 31$, reject.

Swap C and F. Result: $D - E - F - C - A - B - D$, length $\ell = 36$, reject.

Swap F and A. Result: $D - E - C - A - F - B - D$, length $\ell = 26$, reject.

Swap A and B. Result: $D - E - C - F - B - A - D$, length $\ell = 15$, accept. This solution becomes the new benchmark from which the swap steps continue.

6 Location Models

This chapter introduces the basic ideas of location models. We first provide a short introduction to the subject and enumerate some of its major components. This is followed by a detailed discussion of the major classes of location models.

6.1 The Major Elements of Location Problems

The origins of location theory are shrouded in history. The first to discuss location models (and here, we use the term in the widest possible sense), were mathematicians. One of the famous location-based puzzles (regarding the point in the triangle from which the sum of distances to the triangle's vertices is minimal) were investigated and solved by Torricelli and Fermat in the 17th century. The geographer von Thünen wrote about his famed "von Thünen circles" regarding the location of economic activities around a central place, and the German geographer Weber wrote a treatise concerning location models early in the 20th century. Hakimi (1964) introduced location models to the field of operations research. We will encounter his famous theorem below. Since then, thousands of contributions have been made by researchers from fields as diverse as mathematics, computer science, geography, business administration, and economics.

While many of us have a pretty good idea what location problems are, let us take a step back and examine its major components. The three main components of every location model are space, customers and facilities. Supplies exist at the facilities, demand occurs at the customer sites, and the goods are "somehow" transported from the facilities to the customers. Let us now look at these components in some detail.

We distinguish between two major classes of location models, those that occur in the plane (or sometimes in three-dimensional space), and those that occur in transportation networks. While not altogether correct, location models in the plane tend to look at the problem from a macro point of view, while those models in transportation networks investigate the scenario from a micro perspective. Location problems in the plane are called *continuous location models* (as the facilities that are to be located can be sited anywhere in the space under consideration) in

H.A. Eiselt and C.-L. Sandblom, *Operations Research: A Model-Based Approach*,
DOI 10.1007/978-3-642-10326-1_6, © Springer-Verlag Berlin Heidelberg 2010

contrast to *discrete location models*, many of which occur in networks. In discrete location models facilities can be located only at a finite number of points.

These two classes of problems are also different from the way the variables are defined: determining the location of the variables in continuous models in the two-dimensional plane requires the definition of variables (x_1, y_1), (x_2, y_2), … that symbolize the coordinates of the facilities 1, 2, … . On the other hand, in network models, we need a variable y_j for each potential location that will assume a value of one, if we actually do locate at site j, and 0 if we do not. This places continuous location models into the field of linear or nonlinear optimization, while network location models are typically formulated and solved as integer programming problems.

As far as parameters go, we assume that the locations of our customers are known, and so are their demands, which are commonly referred to as weights. Formally, customer i (or, equivalently, customers at demand point i) are assumed to have a weight of w_i. In the 2-dimensional (Euclidean) plane, customer i is assumed to be located at a point with coordinates (a_i, b_i), while in a network, customer i can be found at node n_i. Note that the points at which customers are located typically represent either census units, towns, or other customer agglomerations.

Assume now that transportation takes place in the plane. While there exist many different metrics and gauges, most authors use either rectilinear (or Manhattan) distances, Euclidean or straight-line distances, or, sometimes, squared Euclidean distances. Suppose that a customer is located at a point with coordinates (a, b), while the facility is located at (x, y). The *rectilinear* or *Manhattan distance* between the customer and the facility is then defined as $|a - x| + |b - y|$, i.e., it is assumed that all movements take place parallel to the axes of the system of coordinates. The *Euclidean distance* between the customer and the facility is defined as $\sqrt{(a-x)^2 + (b-y)^2}$, and the *squared Euclidean distance* between these two points is $(a-x)^2 + (b-y)^2$. As an example, the Manhattan, Euclidean, and squared Euclidean distances between the points (2, 3) and (7, 1) are 7, $\sqrt{29} \approx$ 5.3852, and 29, respectively.

In network models, it is common practice to assume that movements take place on the shortest path between the customer and the facility. This means that the shortest path algorithms (see Section 5.3 of this book) will typically have to be applied before approaching a location model. In general, we will use the term "distance," by which we may mean the actual mileage between points or any other disutility such as time or costs.

Other components of location problems include the following:

- the number of facilities (which may be fixed by the decision maker or may be endogenous to the model),

- the magnitude of the demand (which may be fixed, e.g., for essential goods, or may depend on the proximity of the facility to the customer),
- the way customers are assigned to facilities (they either choose themselves, as is the case in the context of retail facilities, which are customer choice models, or firms allocate customers to their facilities in allocation models, as is the case in deliveries from warehouses),
- the type of deterministic or probabilistic parameter we are dealing with,
- the single-level or hierarchical model (hierarchical models often exist in the context of health care, with levels such as doctor, local clinic, and regional hospital), where a higher-level facility can accomplish all tasks a lower-level facility can. A clearly defined referral system is crucial in multi-level models,
- competitive or noncompetitive models (most competitive scenarios use game theory as a tool and have firms compete in location, prices, and quantities).

One main component of any location model is the objective function pursued by the decision maker. Until the mid-1970s, location models assumed that the facilities to be located by the decision maker would be "attractive" to the customers in some sense. That way, proximity to the customers was the main part of the objective. Many facilities, however, do not fall into that mold. Consider, for instance, a grocery store, a facility customers normally would consider desirable to have in their proximity. However, as deliveries are made very early in the morning, the same facility may very well be considered (at least partially) undesirable. Clearly, power plants and landfills will be considered undesirable by most customers. While a customer would attempt to "pull" a desirable facility towards himself, he will attempt to "push" away an undesirable facility. In addition to push and pull models, there are also "balancing" models, in which the decision maker attempts to locate facilities so as to balance the business the individual facilities have. This is often the case in the location of automobile dealerships, motels of a chain, and fast food franchises, which are located so as to avoid cannibalizing demand from its own (other) facilities.

Among the pull objectives, three classes have received most of the attention. The first of these classes of models deals with *covering objectives*. The main idea is to locate facilities, so that as many customers as possible are covered, i.e., within a given distance of any of the facilities. Such objectives are often used for the location of emergency equipment, i.e., ambulances, police stations, and fire stations. The other two classes are *center and median locations*. In both of them, the decision maker locates facilities so as to minimize a measure of distance. In center problems, the planner will attempt to locate facilities, so that the largest distance to customers is as small as possible. This objective was justified (in some sense) by Rawls's "Theory of Justice," according to which the quality of a solution is determined by the lowest quality of any of its components. While such

objectives may be justified in the context of reliability systems, it takes an extreme view and is strongly biased towards outliers. Median problems, on the other hand, are probably the largest class in location models. Their objective is the minimization of the weighted total distance. In other words, the distance between a customer and his closest facility is determined (this can be considered a proxy for costs), which is then multiplied by the magnitude of the customer's demand. Considering the distance as a proxy for the cost of shipping one truckload from the facility to the customer (or vice versa), the customer's demand then denotes the number of truckloads to be shipped. The sum of all of these weighted distances is then a proxy expression for the costs of the transportation of the goods to all customers.

Applications of location problems range from the location of schools to warehouses and church camps, equipment for the removal of oil spills, warning sirens in towns in case of floods, hurricanes, or tsunamis; bottling plants, landfills, newspaper transfer points, sewage treatment plants, and many other facilities. Other applications include nonphysical spaces. An example are brand positioning models, in which each dimension of the space represents a feature that is deemed relevant to customers. The features of each product will then determine its location in this feature space. Similarly, each (potential) customer can be represented in the same space by his ideal point, i.e., the point that has the features most preferred by the customer. Assuming that each customer purchases the brand closest to his ideal point, it is then possible to determine the estimated sales of each product. then it is also possible to relocate, i.e., redesign, a brand so as to maximize the amount of demand it captures.

6.2 Covering Problems

The idea behind covering models is to locate facilities that provide some service required by customers. If customers are positioned within a certain predefined critical distance D from any of the facilities, then they are considered served or "covered." Two objectives for the location of facilities are to either cover all customers in the network with the smallest number of facilities or, alternatively, to cover as many customers as possible with a given number of facilities. Typical examples of applications of covering models are found when emergency facilities are to be located. Whether the facilities in question are fire stations, ambulances, police cruisers, or any similar "facilities," the objective is to maximize protection. However, measuring protection is very difficult. In order to find an expression for "protection," we would need to know the value of responding to an emergency from different distances or times. We can safely assume that an increase in the time to respond to an area for fire protection will mean that the fire has a greater chance to spread and it may reduce the chance to save property or a life if one is in jeopardy. A similar argument applies to other types of protection. Unfortunately, measurements of the value of protection are nearly impossible to make. In the case of fire protection, standards of service have been suggested by the Insurance Services Office that, when met, virtually guarantee an adequate level of protection

or, at least, the lowest premiums for fire protection. For fire services the most commonly used surrogate measure of performance is whether service can be provided to areas within a prespecified distance or time. Here, we restrict ourselves to covering problems on networks.

6.2.1 The Location Set Covering Problem

One of the first models that was developed to site emergency service facilities that incorporates a maximum service distance standard is the *Location Set Covering Problem* (*LSCP*) introduced in the early 1970s. The objective of the *LSCP* is to locate the smallest number of facilities such that each demand node is "covered" by one or more facilities. A demand node is said to be covered, if there is a facility within a prespecified distance from the demand node. In the context of the location of a fire hall, the decision maker could specify that a building is covered (or sufficiently protected), if it is within 5 miles (or 7 minutes or some similar measure) of the fire hall.

If we assume that the cost to purchase land and build a facility is roughly the same for all nodes (or is small in comparison to maintaining each of the needed fire crews), then the objective of using the least number of facilities is equivalent to minimizing the cost of providing service to all demand.

Throughout this chapter, we will use the subscript i to denote customers, while the subscript j is employed to denote facilities. For simplicity, we also assume that all customers and facilities will be located at the nodes of the network. We can then define d_{ij} as the shortest distance (or time, cost, or any other disutility) between the demand node n_i and the facility site at node n_j. In addition, let the service standard D denote the maximal service distance or time as specified by the decision maker. We can then define a *covering matrix* $\mathbf{C}(D) = (c_{ij})$, so that $c_{ij} = 1$, if customers at node n_i can be covered from a facility located at node n_j, and 0 otherwise.

While it is possible to solve location set covering problems as integer programming problems by using one of the standard commercial codes, it is usually a good idea to apply a set of reduction rules that typically allow the user to dramatically reduce the size of the problem and accelerate its solution. In quite a few cases, the reduction rules actually solve the problem without us having to resort to an integer programming problem at all.

The principles of the reduction technique are easily described, assuming that a covering matrix $\mathbf{C}(D)$ is available.

- *Essential Column*. A column j is *essential*, if there exists a unit row (i.e., a row with all zeroes and only a single "1") with the "1" in column j. If an essential column exists, then a facility must be located at the node n_j.

- *Dominated Column.* A column j *dominates* a column k, if all elements in column j are at least as large as those in column k. If a column is dominated, it can be deleted.

- *Dominated Row*. A row i is said to *dominate* a row ℓ, if all elements in row i are less than or equal to those in row ℓ. If a row is dominated, it can be deleted.

The rationale behind the three rules is as follows.

In case of a unit row in, say, row i, with the "1" in position "j," customer i can only be covered by locating a facility at node n_j. Since it is required to cover all customers, we must locate a facility at node n_j. Given that, we can then delete column j (there is no need to locate another facility at the same node) and all customers that have a "1" in column j, as the facility at node n_j covers them and we are not concerned with them anymore.

In case one column (i.e., potential facility location), say column j, has ones wherever another column, say column k, does, and possibly a few more, then a facility located at node n_j will cover all customers that a facility at node n_k could, and possibly a few more. As a result, we would never locate a facility at node n_k, and hence it can be deleted.

Finally, we say that row i dominates row ℓ, if all entries of row i are less than or equal to those of row ℓ. This means that if a facility covers customer i, then it also covers customer ℓ, but not necessarily vice versa. Loosely speaking, this implies that customer i is more difficult to cover, so that we can delete the customer who is easier to cover.

The individual rules can be applied repeatedly and in random order. If the reduction terminates with no matrix remaining, an optimal solution has been identified, and the problem is solved. If this is not the case, the reduced problem will then have to be solved by integer programming.

As an example, consider the network in Figure 6.1.

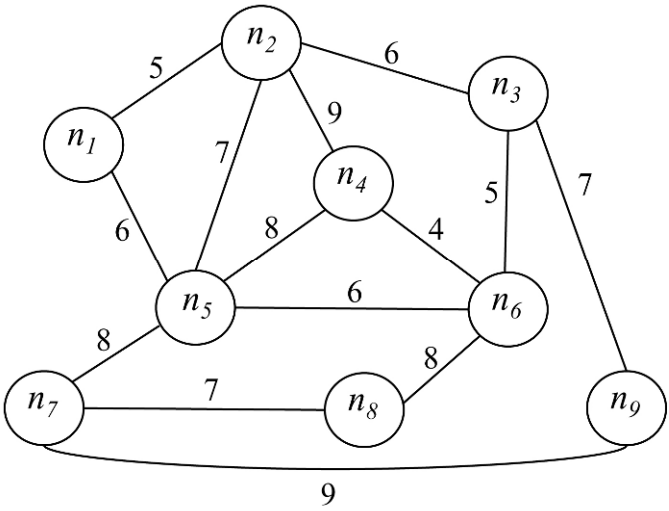


Figure 6.1

The matrix of shortest distances is

	n_1	n_2	n_3	n_4	n_5	n_6	n_7	n_8	n_9
n_1	0	5	11	14	6	12	14	20	18
n_2	5	0	6	9	7	11	15	19	13
n_3	11	6	0	9	11	5	16	13	7
n_4	14	9	9	0	8	4	16	12	16
n_5	6	7	11	8	0	6	8	14	17
n_6	12	11	5	4	6	0	14	8	12
n_7	14	15	16	16	8	14	0	7	9
n_8	20	19	13	12	14	8	7	0	16
n_9	18	13	7	16	17	12	9	16	0

Given a covering distance of $D = 10$, the covering matrix is then

	n_1	n_2	n_3	n_4	n_5	n_6	n_7	n_8	n_9
n_1	1	1	0	0	1	0	0	0	0
n_2	1	1	1	1	1	0	0	0	0
n_3	0	1	1	1	0	1	0	0	1
n_4	0	1	1	1	1	1	0	0	0
n_5	1	1	0	1	1	1	1	0	0
n_6	0	0	1	1	1	1	0	1	0
n_7	0	0	0	0	1	0	1	1	1
n_8	0	0	0	0	0	1	1	1	0
n_9	0	0	1	0	0	0	1	0	1

It is apparent that there are no essential columns. As far as columns are concerned, column 2 dominates column 1 which, in turn, can be deleted. Considering the reduced matrix (i.e., the matrix after deleting column 1), we note that row 1 dominates rows 2, 4, and 5, so that rows 2, 4, and 5 can be deleted as well. The reduced matrix is then

	n_2	n_3	n_4	n_5	n_6	n_7	n_8	n_9
n_1	1	0	0	1	0	0	0	0
n_3	1	1	1	0	1	0	0	1
n_6	0	1	1	1	1	0	1	0
n_7	0	0	0	1	0	1	1	1
n_8	0	0	0	0	1	1	1	0
n_9	0	1	0	0	0	1	0	1

We now repeat the procedure. First, there are still no essential columns. As far as dominated columns are concerned, we find that column 3 dominates column 4, but after the dominated column is deleted, there are no more dominated rows. The next reduced matrix is thus

	n_2	n_3	n_5	n_6	n_7	n_8	n_9
n_1	1	0	1	0	0	0	0
n_3	1	1	0	1	0	0	1
n_6	0	1	1	1	0	1	0
n_7	0	0	1	0	1	1	1
n_8	0	0	0	1	1	1	0
n_9	0	1	0	0	1	0	1

We are now unable to find any further essential columns, dominated rows or columns. This means that we now have to solve this reduced problem by way of integer programming. In order to do so, we define binary variables y_j which assume a value of 1, if a facility is located at node n_j, and 0 otherwise. We are then able to formulate the (reduced) *LSCP* as the following integer programming problem.

$$
\begin{aligned}
\textit{LSCP}\text{: Min } z &= y_2 + y_3 + y_5 + y_6 + y_7 + y_8 + y_9 \\
\text{s.t. } & y_2 + y_5 \geq 1 \\
& y_2 + y_3 + y_6 + y_9 \geq 1 \\
& y_3 + y_5 + y_6 + y_8 \geq 1 \\
& y_5 + y_7 + y_8 + y_9 \geq 1 \\
& y_6 + y_7 + y_8 \geq 1 \\
& y_3 + y_7 + y_9 \geq 1 \\
& y_1, y_2, \ldots, y_9 \geq 0.
\end{aligned}
$$

The objective function minimizes the number of facilities that are located, and the constraints ensure that at least one facility is located within reach of the (remaining) customers 1, 3, 6, 7, 8 and 9.

First, we solve the problem as a linear programming problem, as often the solution is integer without us requiring it, a property sometimes referred to as "integer friendly." In this example, the solution of the linear programming problem is $\bar{y}_2 = ½$, $\bar{y}_3 = ¼$, $\bar{y}_5 = ½$, $\bar{y}_6 = ¼$, $\bar{y}_7 = ¾$, and $\bar{y}_8 = \bar{y}_9 = 0$. The value of the objective function for this solution is $z = 2¼$. Including zero-one conditions for all variables results in the optimal solution $\bar{y}_5 = \bar{y}_6 = \bar{y}_7 = 1$ and all other variables zero, i.e., facilities should be located at the nodes 5, 6, and 7. However, this solution is not unique: other optimal solutions with three facilities locate them at 2, 3, and 7, or at 3, 5 and 8, or at 2, 3 and 8.

In order to further explore the problem, assume now that the distance standard is changed to $D = 9$. The covering matrix is actually identical to that found for $D = 10$. Suppose now that $D = 8$. The coverage matrix is then

	n_1	n_2	n_3	n_4	n_5	n_6	n_7	n_8	n_9
n_1	1	1	0	0	1	0	0	0	0
n_2	1	1	1	0	1	0	0	0	0
n_3	0	1	1	0	0	1	0	0	1
n_4	0	0	0	1	1	1	0	0	0
n_5	1	1	0	1	1	1	1	0	0
n_6	0	0	1	1	1	1	0	1	0
n_7	0	0	0	0	1	0	1	1	0
n_8	0	0	0	0	0	1	1	1	0
n_9	0	0	1	0	0	0	0	0	1

There is no essential column. However, column 2 dominates column 1, and column 3 dominates column 9. As far as rows are concerned, row 1 dominates rows 2 and 5, row 9 dominates row 3, and row 4 dominates row 6. Column 3 is essential as it is the only facility location that can cover customers at node n_9, so that we must locate a facility at n_3. Column 3 and row 9 can then be deleted. This leaves us with the reduced matrix

	n_2	n_4	n_5	n_6	n_7	n_8
n_1	1	0	1	0	0	0
n_4	0	1	1	1	0	0
n_7	0	0	1	0	1	1
n_8	0	0	0	1	1	1

Again, there is no essential column. However, column 5 dominates column 1 and 4, and columns 7 and 8 are equal, so that we retain either one. Row 1 dominates rows 4 and 7, leaving us with

	n_5	n_6	n_7
n_1	1	0	0
n_8	0	1	1

Now row 1 is a unit row, so that column 5 is essential. After removing it, columns 6 and 7 are identical, so that a facility can be located at either site. In summary, we have now located facilities at n_3, n_5, and one of n_6, n_7, or n_8. In other words, even with a distance standard of $D = 8$, three facilities are sufficient to cover all customers. Notice that it was not necessary to solve an explicit integer programming problem at all, the reduction rules alone were sufficient to solve the problem.

If the distance standard is reduced further to $D = 7$, four facilities are necessary to cover all customers. These facilities will have to be located at n_2, n_3, n_6, and n_7, or at n_3, n_5, n_6, n_7, or at n_2, n_3, n_6, n_8, or at n_3, n_5, n_6, n_8.

A further reduction of the distance to $D = 6$ results in five facilities being required to cover all customers. The facilities will be located at n_6, n_7, n_8, n_9, and either n_1 or n_2.

In case the distance standard is set to $D = 5$, we need six facilities to cover all customers. The facilities will be located at n_1, n_5, n_6, n_7, n_8, and n_9.

For a distance standard of $D = 4$, we need no less than eight facilities. In particular, there will be a facility at each node, except for either node n_4 or n_6.

Once the distance standard is $D < 4$, there must be a facility at each node.

On the other hand, increasing the distance standard from the original $D = 10$ to $D = 11$, we can determine that only two facilities are needed to cover all customers. These facilities will be located at n_5 and n_7.

Finally, for any distance standard $D \geq 14$, only a single facility is needed to cover all customers. This facility will have to be located at the node n_6. Note that this solution is unique.

The results are summarized in Figure 6.2, which has the distance standard on the ordinate and the number of facilities required to cover all nodes at the abscissa.

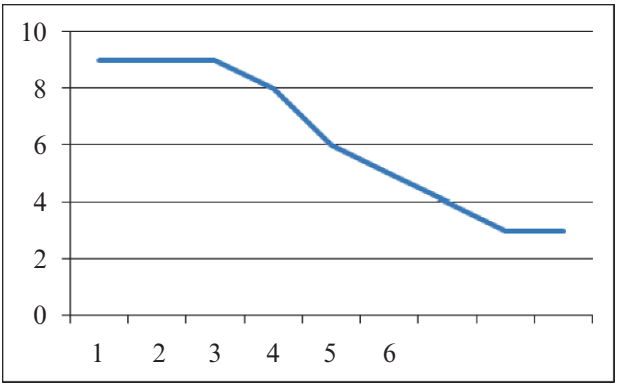


Figure 6.2

6.2.2 The Maximal Covering Location Problem

In contrast to the location set covering problem discussed above, the *Maximal Covering Location Problem* (*MCLP*) does not attempt the task to cover *all* customers. Given a fixed number of p facilities, the task is to locate these facilities so as to cover the largest possible number of customers. In addition to the parameters defined in the previous section, we also need w_i, which denotes the number of customers (or the magnitude of the demand) at node n_i.

As an illustration, consider again the example of the previous section with the assumption that exactly two facilities are to be located. Furthermore, let the covering distance be $D = 10$. The weights (i.e., the number of customers at a node) are given as follows:

n_1	n_2	n_3	n_4	n_5	n_6	n_7	n_8	n_9
120	160	100	110	130	140	190	220	200

In order to formulate the problem, we not only need the binary location variables y_j, but we also need coverage variables x_i, $i = 1, \ldots, n$. A coverage variable x_i assumes a value of one, if a customer at node n_i is covered by at least one facility, and zero otherwise. The main reason for these additional variables is to avoid double counting. The formulation of our problem is then

$$\text{Max } z = 120x_1 + 160x_2 + 100x_3 + 110x_4 + 130x_5 + 140x_6 + 190x_7 + 220x_8 + 200x_9$$

$$\text{s.t. } y_1 + y_2 + y_3 + y_4 + y_5 + y_6 + y_7 + y_8 + y_9 = 2$$
$$x_1 \leq y_1 + y_2 + y_5$$
$$x_2 \leq y_1 + y_2 + y_3 + y_4 + y_5$$
$$x_3 \leq y_2 + y_3 + y_4 + y_6 + y_9$$
$$x_4 \leq y_2 + y_3 + y_4 + y_5 + y_6$$

$x_5 \leq y_1 + y_2 + y_4 + y_5 + y_6 + y_7$
$x_6 \leq y_3 + y_4 + y_5 + y_6 + y_8$
$x_7 \leq y_5 + y_7 + y_8 + y_9$
$x_8 \leq y_6 + y_7 + y_8$
$x_9 \leq y_3 + y_7 + y_9$
$y_j = 0$ or 1 for all j and $x_i = 0$ or 1 for all i.

Each term in the objective function will count all customers at a node being covered, if and only if the node is covered by a facility. The main reason for the use of the covering variables x_i is to ensure that customers who are covered by more than one facility will not be counted more than once. The first constraint ensures that exactly 2 facilities will be located. The remaining constraints define the coverage of a node. In particular, a node is considered covered, if there is at least one facility within covering distance. In this example, consider the node n_1. This node could potentially be covered from a facility at node n_1, n_2, or n_5. Its covering constraint is $x_1 \leq y_1 + y_2 + y_5$, which ensures that if there is no facility at either n_1, n_2, or n_5, then the right-hand side value of the constraint equals zero, which forces the variable x_1 to assume a value of zero as well. On the other hand, if there exists at least one variable at any one of the three nodes, the right-hand side value of the inequality is at least one, which renders it redundant, as x_1 is defined as a zero-one variable anyway. However, while x_1 could assume a value of either zero or one in such a case, the objective function includes the term $120x_1$ which is part of what is to be maximized. This pushes the value of x_1 to as large a value as possible, so that it will assume a value of "1," whenever possible.

The optimal solution of the problem is $\bar{y}_5 = \bar{y}_7 = 1$, meaning that we will locate facilities at the nodes n_5 and n_7. All coverage variables except x_3 equal one, and the value of the objective function equals $\bar{z} = 1{,}270$. This approach is viable for smaller problems, but may fail for large problems due to the large number of variables (which equals twice the number of nodes in the underlying network). So, rather than solving the problem exactly, we may resort to a heuristic algorithm that may, of course, not necessarily find an optimal solution. The heuristic we use below is of the Greedy type in that it locates one facility at a time and in each step it does so by locating the next facility, so as to maximize the number of additional customers that are covered in that step. Being a heuristic, the myopic Greedy procedure may not necessarily find an optimal solution.

The number of customers that are covered by a single facility located at one of the nodes can then be determined by multiplying the coverage matrix by the vector of weights from the left, i.e., compute $\mathbf{wC}(D)$. The j-th component of the resulting vector indicates the number of customers that will be covered if a facility were to be located at the node n_j. In our example, we obtain

$$[120, 160, 100, 110, 130, 140, 190, 220, 200] \begin{bmatrix} 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \end{bmatrix} =$$

= [410, 620, 710, 640, 850, 700, 740, 550, 490]. The facility that serves most customers is one that is located at n_5, from where it covers 850 customers. Since these customers are covered, we can most easily avoid double counting by deleting the rows that belong to nodes, which are already covered by a facility at n_5. In our example, these are the nodes n_1, n_2, n_4, n_5, n_6, n_7. Deleting column n_5 as well (as we have already located a facility there), the reduced coverage matrix is

	n_1	n_2	n_3	n_4	n_6	n_7	n_8	n_9
n_3	0	1	1	1	1	0	0	1
n_8	0	0	0	0	1	1	1	0
n_9	0	0	1	0	0	1	0	1

Multiplying this matrix with the remaining vector of weights [100, 220, 200] from the left, we obtain [0, 100, 300, 100, 320, 420, 220, 300], whose unique maximum is in the position that belongs to n_7. That way, we cover an additional 420 customers and we can now delete the rows that belong to nodes n_8 and n_9 as well as the column of n_7. Since we have now exhausted our resources in the form of two facilities, customers at n_3 will remain unserved and we have succeeded covering a total of 850 + 420 = 1,270 customers by locating the two facilities at nodes n_5 and n_7. This, as we happen to know, is the optimal solution.

Typically, a construction heuristic such as Greedy will be followed by an improvement heuristic. One such improvement heuristic is the swap-interchange improvement heuristic. The idea is simple: in each step one facility is removed from its present location and it is relocated to a site, at which there presently is no facility. If such a move increases coverage, it is accepted, otherwise the search for better solution continues until some stop criterion is satisfied.

We will illustrate this procedure by using again the example shown above. However, since the Greedy heuristic already found an optimal solution, start with some other nonoptimal solution. Suppose that we locate facilities at nodes n_3 and n_6. The two facilities cover customers at the nodes n_2, n_3, n_4, n_6, and n_9 and n_3, n_4, n_5, n_6, and n_8, respectively, so that all customers except those at nodes n_1 and n_7 are covered.

We first try to remove a facility from its present location at n_3 and move it to n_1 instead. Such a move means that we lose coverage of customers at n_2 and n_9 for a loss of 360 demand units, while the facility at the new location will cover customers at n_1, n_2, and n_5 (of whom customers at n_5 were already covered by the facility at n_6), so that we gain 280 new customers. This indicates that our move results in a net loss of 80 customers, so that the move is rejected.

Instead, we may try to relocate a facility from n_3 to n_2. A similar argument will reveal the same loss of 80 customers, so this move is also rejected.

Yet another possibility is to relocate the facility from n_3 to n_4. This move results in a net loss of 200 customers, so again, the move is rejected.

Consider now the relocation of a facility from n_3 to n_5. Again, the loss of customers due to the removal of a facility from n_3 equals 360 customers, while the relocation of the facility at n_5 will cover the previously uncovered customers at n_1, n_2, and n_7 for a gain of 470 customers, so that there is a total net gain of 110 customers. This means that the move is accepted, so that the new solution has facilities located at n_5 and n_6. This process is repeated until no further improvements are possible. Again, the swap method is a heuristic and as such is not guaranteed to find an optimal solution.

We would like to conclude this section with a number of extensions of the basic covering models discussed above. One possibility to deal with outliers (whose coverage is very expensive) is to attempt to cover at least a certain proportion of the population within a prespecified distance or time. For instance, one could attempt to locate fire stations that has 90% of the potential customers within 8 minutes of the station.

Another important issue concerns congestion. Especially when resources are scarce (e.g., ambulances), an important issue is what happens if a service call arrives while the unit is presently busy. A possible way to deal with such situations is the introduction of backup coverage. In other words, the decision maker may try not just to cover each potential customer once, but try to cover a certain proportion of customers more than once. The obvious question then concerns the tradeoff between primary coverage for some people versus backup coverage for others.

6.3 Center Problems

Another classical problem type comprises the so-called *center problems*. The basic idea of all center problems is that they locate facilities so as to minimize the longest distance between a customer and his closest facility. The underlying logic of center problems is based on Rawls's "Theory of Justice," according to which

the quality of a solution depends on the least well-served entity. One of the problems associated with the concept of centers is their exclusive focus on the customer with the longest facility–customer distance. This can lead to highly undesirable outcomes.

All center problems have an objective that minimizes the maximal (i.e., longest) distance between the facility and any customer. In case of problems that involve multiple facilities, the center objective minimizes the maximal distance between any customer and its nearest facility.

Throughout this section, we assume again that all demand is clustered at the nodes of the network (or at given points in the plane). We can then distinguish between *node centers*, which are facility locations for which only the nodes are considered, and *absolute centers*, in which case facilities may be located anywhere on the network (including on arcs) or in the plane.

The need for this distinction is apparent by considering a graph with only a single arc, at whose edges we have the nodes n_1 and n_2. Either node would serve as a single node center (whose critical distance, i.e., the distance from the facility to the farthest customer) equals the length of the arc. However, the point on the graph that minimizes the maximal distance is at the center of the arc, from where the longest distance to either customer is only half the length of the arc.

6.3.1 1-Center Problems

The simplest center problem is the 1-node center problem on a network. The problem is to find a facility location, so as to minimize the maximal distance between the facility and any of the customers. As an illustration, consider again the graph in Figure 6.1 in the previous section.

Recall that the matrix of shortest distances was

	n_1	n_2	n_3	n_4	n_5	n_6	n_7	n_8	n_9
n_1	0	5	11	14	6	12	14	20	18
n_2	5	0	6	9	7	11	15	19	13
n_3	11	6	0	9	11	5	16	13	7
n_4	14	9	9	0	8	4	16	12	16
n_5	6	7	11	8	0	6	8	14	17
n_6	12	11	5	4	6	0	14	8	12
n_7	14	15	16	16	8	14	0	7	9
n_8	20	19	13	12	14	8	7	0	16
n_9	18	13	7	16	17	12	9	16	0

Again, the rows symbolize the customers, while the columns stand for the facilities. If we were to position a facility at node n_1, then the distances between

the customers and the facility would be found in the first column. The longest such distance is then the largest number in the first column, in our case 20.

A simple brute-force enumeration procedure now suggests itself for the determination of a 1-node center on graphs: tentatively locate a facility at a node, determine the distance to the farthest customers, repeat for all potential facility locations, and choose the shortest distance. In other words, determine all column maxima in the matrix, and chose the 1-center facility location as the column with the shortest such distance. In our example, the column maxima are [20, 19, 16, 16, 17, 14, 16, 19, 18], so that the 1-node center is located at node n_6 and the longest facility–customer distance is 14.

Consider now the 1-absolute center problem in the plane with Manhattan distances. This problem has a very simple closed-form solution. Assume that the n customers are located at the points P_1, P_2, …, P_n with coordinates (a_1, b_1), (a_2, b_2), …, (a_n, b_n), and that the facility will be located at a point with the coordinates (x, y). Note that regardless of the number of customers, the problem has only two variables, *viz*., x and y.

In order to solve the problem, we need to define five auxiliary variables. They are

$$\alpha_1 = \max_i \{a_i + b_i\}$$
$$\alpha_2 = \max_i \{-a_i + b_i\}$$
$$\alpha_3 = \max_i \{a_i - b_i\}$$
$$\alpha_4 = \max_i \{-a_i - b_i\}$$
$$\alpha_5 = \max \{(\alpha_1 + \alpha_4), (\alpha_2 + \alpha_3)\}.$$

The 1-absolute center is then located at (x, y) with coordinates

$x = ½(\alpha_3 - \alpha_4)$ and $y = ½(-\alpha_3 - \alpha_4 + \alpha_5)$ with longest distance $z = ½\alpha_5$,

and, alternatively,

$x = ½(\alpha_1 - \alpha_2)$ and $y = ½(\alpha_1 + \alpha_2 - \alpha_5)$ with longest distance $z = ½\alpha_5$.

As a numerical example, consider a problem with six customers, who are located at the points P_1: (0, 0), P_2: (0, 3), P_3: (1, 6), P_4: (4, 5), P_5: (4, 2), P_6: (5, 0). We can then calculate

$\alpha_1 = \max\{0+0, 0+3, 1+6, 4+5, 4+2, 5+0\} = 9,$
$\alpha_2 = \max\{-0+0, -0+3, -1+6, -4+5, -4+2, -5+0\} = 5,$
$\alpha_3 = \max\{0-0, 0-3, 1-6, 4-5, 4-2, 5-0\} = 5,$
$\alpha_4 = \max\{-0-0, -0-3, -1-6, -4-5, -4-2, -5-0\} = 0,$ and
$\alpha_5 = \max\{(9 + 0), (5+5)\} = 10.$

The 1-absolute center facility location is then at

$x = 2½$ and $y = 2½$ with $z = 5$, or, alternatively, $x = 2$, $y = 2$, with $z = 5$.

The Manhattan distances between the individual customers and the facility are 5, 3, 5, 4, 2, and 5 in case the facility is located at (2½, 2½), and they are 4, 3, 5, 5, 2, 5 in case the facility is located at (2, 2). It is apparent that the longest customer-facility distance equals 5 in both cases.

6.3.2 *p*-Center Problems

This subsection considers node *p*-center problems in networks. The objective of *p*-center problems is to locate a given number *p* of facilities, so that the longest distance between any customer and its closest facility is as small as possible. While, from a theoretical point of view, *p*-center problems are difficult, a simple bisection method can be employed to solve *p*-center problems as a sequence of covering problems.

The method commences with an initial guess of the covering distance *D*. We then determine the smallest number of facilities required to cover all nodes given *D*. The model to do so is, of course, the location set covering problem. Suppose that the number of facilities required to cover all customers is $p(D)$. If $p(D) > p$, then *D* has to be revised upward; if $p(D) \leq p$, then *D* can be decreased. This process continues until a covering distance is found that cannot be reduced any further without requiring additional facilities.

The initial interval $[\underline{D}; \overline{D}]$ must be sufficiently large to include the optimal solution. An obvious choice of the lower bound is $\underline{D} = 0$, while $\overline{D} = (n-1)\max\{d_{ij}\}$ with the maximum taken over all edges in the network, is an upper bound on the length of the longest path in the network. The reason is that no path in the network can have more than $(n-1)$ edges, and the length of each arc is no more than that of the longest arc. Once lower and upper bounds on the covering distance are determined, we apply a bisection search. This process can be described as follows.

First we bisect the present interval $[\underline{D}, \overline{D}]$ and choose a covering distance D in the center of this interval. In particular, we choose $D := \lfloor \tfrac{1}{2}(\overline{D} + \underline{D}) \rfloor$ We then solve a location set covering problem with the service standard D. The resulting number of facilities required to cover all nodes is denoted by $p(D)$. If $p(D) \le p$, then we reduce the upper bound of the interval $[\underline{D}, \overline{D}]$ to $\overline{D} = D$, otherwise we move up the lower part of the interval to $\underline{D} = D + 1$. Whenever the lower and upper bounds of the interval are equal, we have found the optimal solution, in which case the last-solved covering problem provides the location pattern of the facilities. As long as $\overline{D} \neq \underline{D}$, we determine a new covering distance D on the basis of the interval $[\underline{D}, \overline{D}]$ with a new lower or upper bound. Then a new covering distance D is computed as the center point of this interval and the procedure is repeated.

As a numerical illustration, consider the network in Figure 6.3 and assume that $p = 3$ facilities are to be located.

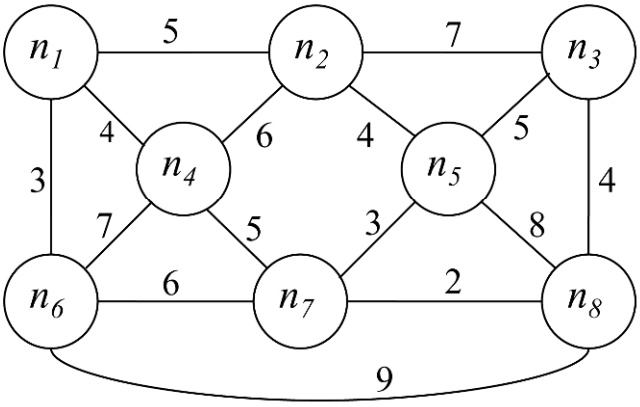


Figure 6.3

We initialize the computations with a lower bound of $\underline{D} = 0$ and an upper bound of $\overline{D} = 7(9) = 63$. Bisecting the interval results in the trial value of $D = 31$, for which the set covering problem has a solution of $p(D) = p(31) = 1$, i.e., a single facility, located anywhere, will cover all the nodes. As $p(31) = 1 < 3 = p$, we set $\overline{D} := 31$.

The results of the subsequent iterations are summarized in Table 6.1.

Table 6.1: Bisection search to find a p-center solution in the example

Iteration #	$\underline{D}$	$\overline{D}$	D	$p(D)$	Location
1	0	63	31	1	anywhere
2	0	31	15	1	anywhere
3	0	15	7	2	e.g., (n_1, n_8), or (n_2, n_7), or (n_3, n_6)
4	0	7	3	5	n_1, n_3, n_4, then n_7, and one of n_1 and n_6
5	4	7	5	2	(n_1, n_5), or (n_1, n_8)
6	4	5	4	3	e.g., n_1 and n_5, and one of n_3 and n_8
7	4	4	4		Stop, optimal.

6.4 Median Problems

As opposed to the center problems with their minimax objective discussed in the previous section, this section is devoted to median problems which have minisum objectives. In other words, they will locate facilities, so as to minimize the sum of distances to the customers. This feature makes this type of objective amenable to applications in the public and the private sector. Consider, for instance, the location of a public facility such as a library. The municipal planner will attempt to make the library as accessible as possible to all of its potential patrons. This may be done by minimizing the average distance between the library and its customers. It is not difficult to demonstrate that as long as the magnitude of the demand remains constant, minimizing the sum of facility-customer distances is the same as minimizing the average facility-customer distance.

6.4.1 Minisum Problems in the Plane

Throughout this section, we assume that the task is to locate a single new facility anywhere in the plane. The n customers are assumed to be located at points P_1, P_2, …, P_n with coordinates (a_i, b_i) and their demand is denoted by the weight w_i. The task at hand is now to locate a facility, whose coordinates are the variables (x, y). Note that regardless of the number of customers in the problem, the model has no more than two variables. As discussed above, the objective is to minimize the weighted sum of customer-facility distances, a proxy of the total cost of the transportation. This type of problems (in the plane) is frequently referred to as *Weber problem* in reference to the work by the German economist-turned sociologist Weber (1868 – 1958) that culminated in the publication of his book in 1909.

The simplest problem occurs when rectilinear distances are used. The objective function is then separable, meaning that it is possible to optimize one variable at a time. It turns out that the actual distances are irrelevant, it is only important how the customers are located in relation to each other, but not how far from each

other. Actually, the procedure is very simple. We first scan the customers from left to right (or right to left) along the abscissa and add their weights, until the sum of weights for the first time matches or exceeds half of the total weight. We then repeat the process by scanning the facilities and adding their weights from top to bottom (or bottom to top), until again the reach or exceed half the total weight for the first time. The combination of the two coordinates is the location that minimizes the sum of weighted rectilinear distances.

In order to illustrate the procedure, consider twelve customers P_1, P_2, …, P_{12}, whose locations are (a_1, b_1), (a_2, b_2), …, (a_{12}, b_{12}) and their weights are w_1, w_2, .., w_{12}. The numerical values of the coordinates and weights are shown in Table 6.2.

Table 6.2: Locations and weights of customers in the example

Point P_j	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9	P_{10}	P_{11}	P_{12}
a_i	2	1	2	4	5	4	3	5	8	10	9	7
b_i	1	4	5	2	4	6	7	9	9	10	5	2
w_i	20	20	10	10	50	30	40	60	70	80	50	60

The arrangement of customers can be visualized in Figure 6.4.

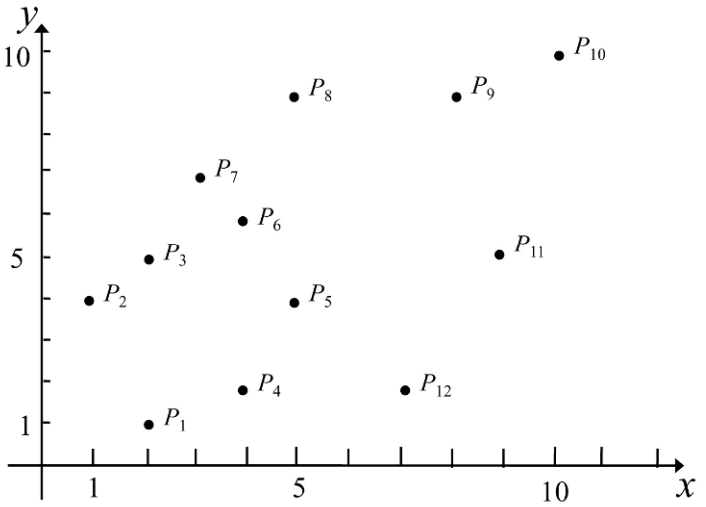


Figure 6.4

First scan the abscissa. From left to right, customers can be rearranged as P_2, then P_1 and P_3, then P_7, followed by P_4 and P_6, then P_5 and P_8, and so forth. Adding up weights in that order results in 20 + (20 + 10) + 40 + (10 + 30) + (50 + 60) + … + 80. Given that the total weight, i.e., the total demand of all customers, equals 500,

we need to add up these values until, for the first time, they reach or exceed the value of 500/2 = 250. This happens for customer P_{12} at the value $x = 7$. (The same result is achieved if we were to add up values starting from the right). This procedure is now repeated for the ordinate by adding up weights from either bottom to top or top to bottom. From the bottom, we add 20 + (10 + 60) + (20 + 50) + (10 + 50) + 30, at which point we have reached the value of 250. This occurs for customer P_6 at $y = 6$, so that this is the optimal coordinate.

However, since the critical value of 250 was achieved exactly, there are alternative optimal solutions. If we had added up weights from top to bottom, we would have achieved the value of 250 for customer P_7 at the coordinate $y = 7$, which is also optimal. As a matter of fact, not only are the two points $(x, y) = (7, 6)$ and (7, 7) optimal, all points with $x = 7$ and y between 6 and 7 are optimal. The value of the objective function can then be determined by computing the customer-facility distances for all customers, multiplying them by the appropriate weights, and adding them up. In our example, for the optimal location at (7, 6) we obtain the total weighted distance 20(10) + 20(8) + 10(6) + 10(7) + 50(4) + 30(3) + 40(3) + 60(1) + 70(4) + 80(7) + 50(1) + 60(4) = 2,090.

Consider now the same example as before, but with the weights of P_5 and P_{12} interchanged, i.e., $w_5 = 60$ and $w_{12} = 50$. Now the optimized location is between 5 and 7 on the abscissa, and between 6 and 7 on the ordinate, so that *all* locations in the rectangle with the corners at (5, 6), (5, 7), (7, 7), and (7, 6) are optimal.

Another model works with the same scenario and also minimizes the sum of weighted distances. However, it uses squared Euclidean distances instead of rectilinear distances. Again, the objective function turns out to be separable, so that we are able to optimize for our variables x and y separately. Since these variables are continuous, we can take partial derivatives, set them equal to zero, and solve for the variables x and y. This process results in an optimal solution in which the x-coordinate of the facility equals the sum of weighted x-coordinates of the customers divided by the total weight, and similar for the y-coordinates. The solution obtained by using squared Euclidean distances is the *center-of-gravity* of the given points. As a numerical illustration, consider again the above example with data shown in Table 6.1. Here, we obtain the optimal facility location at $\bar{x} = \frac{2(20)+1(20)+\cdots+7(60)}{20+20\cdots+60} = 3{,}140/500 = 6.28$. The optimal y-coordinate is determined in a similar fashion as $\bar{y} = 3{,}170/500 = 6.34$. Note the difference between this solution and that obtained by using rectilinear distances: despite the strong pull of the customers with large weights in the Northeast corner, the solution that uses squared Euclidean distances locates the facility more towards the Southeast than the solution that uses rectilinear distances. This is due to the fact that by squaring distances, long distances receive a heavy emphasis and the minimization function will try to avoid them. In that sense, a minisum objective with squared Euclidean

distances will find a solution that includes features of the usual minisum, and also those of the minimax objective.

Using Euclidean distances, we can apply a similar approach. However, the objective function is no longer separable. Setting partial derivatives to zero, we obtain the optimality conditions

$$x = \frac{\sum_{i=1}^{n} \frac{w_i a_i}{\sqrt{(a_i - x)^2 + (b_i - y)^2}}}{\sum_{i=1}^{n} \frac{w_i}{\sqrt{(a_i - x)^2 + (b_i - y)^2}}}, \quad \text{and} \quad y = \frac{\sum_{i=1}^{n} \frac{w_i b_i}{\sqrt{(a_i - x)^2 + (b_i - y)^2}}}{\sum_{i=1}^{n} \frac{w_i}{\sqrt{(a_i - x)^2 + (b_i - y)^2}}}.$$

Notice that in these two relations, the variables x and y appear on the left-hand sides as well as on the right-hand sides. A "trick" first devised by Weiszfeld in 1937 is to use these relations in an iterative procedure. It starts with a guess (x, y), inserts it on the right-hand side of the equations, computes new values for x and y on the left-hand side, uses these values on the right-hand side to compute new values on the left, and so forth.

As a starting point, it may be a good idea to use the center-of-gravity or the solution given rectilinear distance. In order to demonstrate the *Weiszfeld method*, consider the following numerical

Example: There are three customers at the points P_1, P_2, and P_3 with coordinates (0, 0), (3, 0), and (0, 5), respectively. The demands of the customers are $w_1 = 30$, $w_2 = 20$, and $w_3 = 40$. As a starting point, we use $(x, y) = (5, 5)$, even though this is quite an unreasonable initial point. The reason is that the optimal solution will always be located in the triangle formed by the customers, and our starting point is not. We have chosen this point to demonstrate that normally—there are counterexamples, though—this technique converges very quickly. Before starting, note that using rectilinear distances in this example results in an optimal facility location at (0, 0), while squared Euclidean distances will locate the facility at (⅔, 2⅔).

Given a starting point with coordinates $x = 5$ and $y = 5$, we compute the next trial point at

$$x = \frac{\frac{0(30)}{\sqrt{(0-5)^2 + (0-5)^2}} + \frac{3(20)}{\sqrt{(3-5)^2 + (0-5)^2}} + \frac{0(40)}{\sqrt{(0-5)^2 + (5-5)^2}}}{\frac{30}{\sqrt{(0-5)^2 + (0-5)^2}} + \frac{20}{\sqrt{(3-5)^2 + (0-5)^2}} + \frac{40}{\sqrt{(0-5)^2 + (5-5)^2}}} \approx .6982,$$

and

$$y = \frac{\dfrac{0(30)}{\sqrt{(0-5)^2+(0-5)^2}} + \dfrac{0(20)}{\sqrt{(3-5)^2+(0-5)^2}} + \dfrac{5(40)}{\sqrt{(0-5)^2+(5-5)^2}}}{\dfrac{30}{\sqrt{(0-5)^2+(0-5)^2}} + \dfrac{20}{\sqrt{(3-5)^2+(0-5)^2}} + \dfrac{40}{\sqrt{(0-5)^2+(5-5)^2}}} \approx 2.5068.$$

Given the new trial point $(x, y) = (.6982, 2.5068)$—which, incidentally, is quite similar to the center-of-gravity determined earlier—we can then compute the next trial point in similar fashion. It turns out to be $(x, y) = (.5366, 2.3512)$, which is quite close to the previous solution. This procedure terminates whenever some stop criterion is satisfied, e.g., there is only a small change in the solution or the value of the objective function.

A good tool that allows us envisage the forces that determine the solution is the *Varignon frame* named after the French mathematician Varignon, 1654-1722. Imagine a board, in which holes have been drilled at the points at which customers are located. One string is fed through each hole and all strings are tied together in one knot. Below the board, weights are attached to the strings, such that the weights are proportional to the demand of the respective customer. Given only gravity and in the absence of friction, the knot will then settle at the optimal point. A graph of the Varignon frame is shown in Figure 6.5.

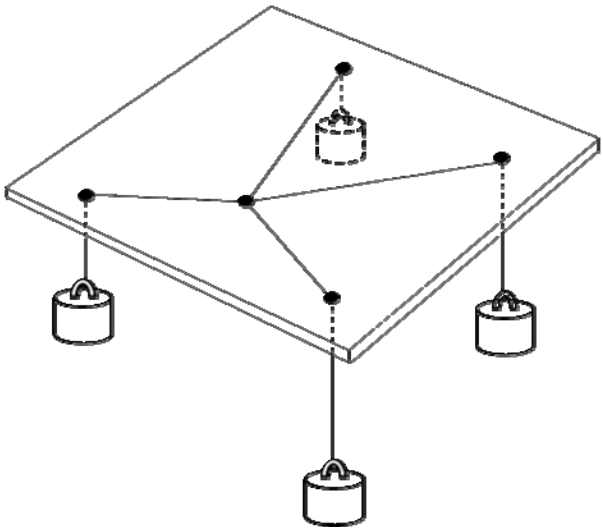


Figure 6.5

6.4.2 Minisum Problems in Networks

Consider now the location of a single new facility on a network. The facility location that minimizes the sum of weighted customer–facility distances is usually referred to as a 1-*median*. An obvious extension of the 1-median problem is the *p-median problem* that locates p facilities so as to minimize the sum of distances between all customers and their respective closest facilities.

The main result we will use is due to Hakimi (1964), who proved the following

Theorem 6.1: At least one optimal solution of a p-median problem is located at a node.

Given this result, we no longer have to search for an optimal solution anywhere on an arc, but can restrict ourselves to the nodes of the network. In that sense, this theorem is reminiscent of Dantzig's corner point theorem that also restricted the set of possible locations from an infinite set to a finite set.

First consider the case, in which the decision maker's task is to determine a 1-median in a network. We assume that the matrix of shortest paths has already been determined. Assuming that the matrix $\mathbf{D} = (d_{ij})$ includes the lengths of the shortest paths between all pairs of nodes, and the vector $\mathbf{w} = (w_i)$, we now use Hakimi's theorem and compute the costs for all potential facility locations at the nodes of the network. This can be accomplished by vector-matrix multiplication. In other words, determine the vector $\mathbf{wD}$, which includes in its j-th element the total transportation cost (i.e., weighted distance) from all customers to a facility located at node j. As an illustration consider the following

Example: The graph in Figure 6.6 includes the single-digit node-to-node (direct) distances next to its edges and double-digit weights next to its nodes.

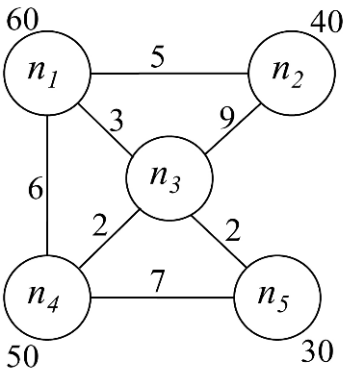


Figure 6.6

The matrix of shortest path distances is then

D:

	n_1	n_2	n_3	n_4	n_5
n_1	0	5	3	5	5
n_2	5	0	8	10	10
n_3	3	8	0	2	2
n_4	5	10	2	0	4
n_5	5	10	2	4	0

while the vector of demands is **w** = [60, 40, 20, 50, 30]. Multiplying the vector and the matrix results in **wD** = [660; 1,260; 660; 860; 940]. This means that if we were to locate a facility at, say node n_4, then our total transportation costs would be 860. Choosing the minimum in this cost vector reveals that we should either locate at node n_1 or at node n_3; in both cases the total transportation costs are 660. In order to choose between the two alternatives, the decision maker may use secondary criteria.

Arguably, *p*-median problems are among the most researched location models in practice. Since they are difficult to solve, most large-scale problems will be solved by heuristic algorithms. This section will describe one construction heuristic and two improvement heuristics.

As usual, the Greedy heuristic works in sequential fashion. In the first stage, it computes the weighted transportation costs for tentative locations at all nodes, and then permanently locates a facility at the node that allows the transportation at the lowest cost. This is exactly the same step as that in the exact method that locates a single facility, i.e., the 1-median on a network.

Suppose now that a number of facilities have already been permanently located and the Greedy heuristic attempts to locate an additional facility. This is accomplished by tentatively locating a new facility at one (presently unoccupied) node at a time. Suppose we tentatively locate a new facility at node n_j. The method will then compute the shortest distance between each customer at node n_i and the closest facility that either exists already or is proposed (at n_j). These distances will be collected in the column vector $\mathbf{d}_j$. The weighted distance is then computed by multiplying the weights and these distances, i.e., $\mathbf{wd}_j$. This number expresses the total transportation costs that are incurred if we, in addition to the already existing facilities, locate a new facility at node n_j. This process is repeated for all possible tentative location, and the minimum is chosen. The location of the new facility is then made permanent and the process continues until the desired number of facilities has been located.

We will illustrate this procedure by a numerical

Example: Consider again the network in Figure 6.6 and assume that the task is to locate $p = 3$ facilities, so as to minimize the total transportation costs. The matrix of shortest distances **D** was already determined in the previous section, and so were the vector of weights **w** and the cost vector **wD** = [660; 1,260; 660; 860; 940]. Again, we choose to locate at either node n_1 or at node n_3. Arbitrarily choose node n_1.

We now tentatively locate the second facility at the nodes n_2, n_3, n_4, and n_5. For the trial location at the node n_2, we now have facilities at n_1 and n_2, so that the distances between a customer and the closest of our facilities is found by taking the minima of columns 1 and 2 in the distance matrix. Here, we obtain $\mathbf{d}_2 = [0, 0, 3, 5, 5]^T$, so that $\mathbf{wd_2} = 460$. For the trial location at n_3, the shortest customer–facility distances are found by computing the minimum among columns 1 and 3 in the distance matrix **D**, resulting in $\mathbf{d}_3 = [0, 5, 0, 2, 2]^T$, so that $\mathbf{wd}_3 = 360$. Similarly, we compute $\mathbf{d}_4 = [0, 5, 2, 0, 4]^T$ with $\mathbf{wd}_4 = 360$, and $\mathbf{d}_5 = [0, 5, 2, 4, 0]^T$ with $\mathbf{wd}_5 = 440$. Among those trials, the lowest location costs are found at either node n_3 or at node n_4. Again, we arbitrarily break the tie and choose a facility location at n_3.

We now have facilities permanently located at the nodes n_1 and n_3. In this iteration, we tentatively locate facilities at n_2, n_4, and n_5, one at a time. For a trial location at n_2, we have locations at n_1, n_2, and n_3, so that we determine the shortest customer-facility distances by computing the minima in the first three columns. This results in $\mathbf{d}_2 = [0, 0, 0, 2, 2]^T$ and $\mathbf{wd}_2 = 160$. Similarly, the trial facilities at n_4 and n_5 result in distances and total transportation costs of $\mathbf{d}_4 = [0, 5, 0, 0, 2]^T$ with $\mathbf{wd}_4 = 260$ and $\mathbf{d}_5 = [0, 5, 0, 2, 0]^T$ with $\mathbf{wd}_5 = 300$. The lowest costs are found for the tentative location at n_2, which is now made permanent. Since we have now located all available facilities, the process terminates with facilities located at n_1, n_2, and n_3. The total transportation costs are then 160.

If we had broken the tie for the first facility in the same way and located at n_1, but chose n_4 for the second facility, we would have ended up with facilities located at n_1, n_2, and n_4 for total costs of 140. On the other hand, if we had broken the tie for the location of the first facility in favor of n_3, we would have ended up with facilities located at n_1, n_2, and n_3 with total transportation costs of 160, the same location pattern as that determined earlier. Notice that none of the tie-breaking rules has been proved superior.

As usual, any construction heuristic should be followed by an application of an improvement heuristic. In this chapter, we will describe two techniques. The first heuristic we apply is the so-called *location-allocation heuristic*. Simply put, it alternates between location and allocation steps. In particular, the technique is

initialized with any solution. The better the solution the procedure starts with, the better the solution may be expected to turn out (although not necessarily so). Suppose now that any facility location with the required p facilities has been determined by eyeballing, an application of the greedy method, or any other technique. The first step of the heuristic is then the allocation phase. In it, we simply assign each customer to its closest facility. This results in p clusters, each with a single facility. In the location phase, we then consider one cluster at a time, remove the facility from it, and determine an optimal facility location in it. This new facility location may or may not be equal to the previous location of the facility. This process is repeated for all clusters.

If there has been any change regarding the facility locations in the last step, the procedure is repeated, until there are no further changes.

Example: Consider again the above example and suppose that facilities have been located at n_3, n_4, and n_5. The transportation costs for this location pattern are 500. Allocating each customer to its closes facility results in clusters $\{n_1, n_2, n_3\}$ for the facility at n_3, $\{n_4\}$ for the facility at n_4, and $\{n_5\}$ for the facility at n_5. The location phase of the heuristic method begins by considering the first cluster. The weights of the customers at n_1, n_2, and n_3 are $\mathbf{w}^* = [60, 40, 20]$, and the partial distance matrix for the three nodes is

$\mathbf{D}^*=$

	n_1	n_2	n_3
n_1	0	5	3
n_2	5	0	8
n_3	3	8	0

Computing the total costs for all three potential facility locations in this cluster (i.e., determine a new single facility in this cluster as shown in the previous section), we obtain 260, 160, and 500, so that we choose the node n_2 as the new facility location in this cluster. The other two clusters include only one node each, so that relocation is not possible. Thus the new location pattern includes facilities at the nodes n_2, n_4, and n_5.

Given these facility locations, the new clusters have $\{n_1, n_2\}$ assigned to the facility at n_2, $\{n_3, n_4\}$ assigned to the facility at n_4, and $\{n_5\}$ assigned to the facility at n_5. The first cluster has weights $\mathbf{w}^* = [w_1, w_2] = (60, 40)$ and with a distance matrix of $\mathbf{D}^* = \begin{bmatrix} 0 & 5 \\ 5 & 0 \end{bmatrix}$, we obtain costs of $\mathbf{w}^*\mathbf{D}^* = [200, 300]$, so that the facility will be relocated to n_1 with a partial cost of 200. The weight and distance matrix for the second cluster are $\mathbf{w}^* = [w_3, w_4] = [20, 50]$, and the distance matrix is $\mathbf{D}^* = \begin{bmatrix} 0 & 2 \\ 2 & 0 \end{bmatrix}$, so that the transportation costs for the two potential facility locations in

this cluster are **w*D*** = [100, 40], meaning that the facility is again located at node n_4 with a partial cost of 40. The last cluster only includes a single node, so that the facility is located at n_5 in that cluster with partial costs of 0. The total costs of this location arrangement are then 200 + 40 + 0 = 240.

The next iteration starts with the present facility locations at n_1, n_4, and n_5. The allocation phase results in the clusters $\{n_1, n_2\}$, $\{n_3, n_4\}$, and $\{n_5\}$, which are the same clusters as in the previous iteration. This means that the procedure has terminated with facilities located at n_1, n_4, and n_5 with total transportation costs of 240. Notice again that this solution is not optimal.

The second heuristic we describe in this chapter is the so-called *vertex substitution method*, a technique that employs a simple "swap" step. Again starting with any location arrangement of the required p facilities. In each iteration, the method tentatively moves one of the facilities from its present location to an unassigned location. If the swap reduces the total transportation costs, we have a new solution, and the process continues. Otherwise, we continue with another pair of locations. The process terminates, if no swap reduces the costs any further.

Example: Consider again the above example and initialize the method with the facilities located at the nodes n_3, n_4, and n_5. Moving a facility from n_3 to n_1 results in a cost reduction from 500 to 240, so the move is made and the new solution has facilities locate at n_1, n_4, and n_5. Moving a facility from n_1 to n_2 raises the costs to 340, so the move is not made. Moving a facility from n_1 to n_3 increases the costs to 500, and again, the move is not made. The move of a facility from n_4 to n_2 leaves the cost at 240, so a tie-breaking rule must be used. Here, we keep the facilities at their present locations. Moving a facility from n_4 to n_3 increases the cost to 300, so the move is not made. Moving a facility from n_5 to n_2 decreases the costs to 160, so we make the move and have a new location arrangement with facilities located at nodes n_1, n_2, and n_4. The procedure is repeated until no further improvements are possible.

6.5 Other Location Problems

This section is designed to introduce some additional types of location models that have been discussed in the literature. The first such models deals with *undesirable facilities*. While it appears apparent that facilities such as sewage treatment plants, landfills, power plants, or prisons are undesirable to have in the neighborhood of a residential area, things are more subtle than that. As a matter of fact, just about all facilities have desirable and undesirable features. Take, for instance, a hospital. Few people will argue that it would be great to have such a facility nearby, the wailing sirens of ambulances that can and will be turned on at any time of day or night will surely be considered a nuisance. A similar argument applies to facilities such as prisons: whereas few people would like to have them nearby (other than

maybe to visit relatives), the many employees who work in these facilities would not appreciate having them at a great distance from their home.

Modeling of the location of undesirable facilities can be done in two different ways. We either use the standard cost-minimizing objective and define "forbidden regions," in which facilities cannot be located, or we use an objective function that pushes the facilities away from the customers, rather than pulling them towards them as cost-minimizing objectives do. There are a number of problems related to both approaches. Using forbidden regions in networks is easy: if it is not desirable or permitted to locate a facility at a node, we simply do not define a location variable y_j for that node (or, equivalently, set the location variable to zero). The process is much more complicated in the plane, where the forbidden regions must be defined as system of linear or nonlinear inequalities. Using a "push" objective is not a straightforward process, either: first of all, if we optimize on an unconstrained set, such an objective would attempt to push the facilities towards infinity, which is obviously not reasonable. This means that it will be required to define a feasible set, in which all facilities must be located. Again, the objective that maximizes the weighted sum between customers and facilities will tend to have the facilities locate at the border of the feasible set. A good example of this behavior is the location of nuclear power plants in France, many of which have been sited along the Rhine river, i.e., the border with Germany. Another problem is that simply exchanging the "Min" for a "Max" in the objective function will not suffice. The reason is that the usual cost-minimizing objective will automatically assign a customer to his nearest facility, which is reasonable in almost all relevant contexts. Similarly, a maximization objective, the objective will automatically assign a customer to his farthest facility, which does not make sense in most cases: customers are most affected by the nearest undesirable facility (as its effects will be most pronounced on the customer), rather than the farthest facility. In order to ensure that the effects of the closest facility are measured and counted towards the objective (e.g., the overall pollution level on the population), additional constraints are required. They result in fairly large formulations for even small problems, thus necessitating the use of heuristic algorithms.

Another strand of work deals with location models that have "equity" objectives. The idea is to locate facilities, so as to make them as equally accessible to all (potential) customers. Models of this nature have a variety of difficulties associated with them. First of all, it is not obvious what "equity" is. Dictionaries will define it as "fairness," a concept just as vague. The location models in this category all deal with equality, rather than equity. More precisely, they attempt to locate facilities, so as to make the shortest customer-facility distances (i.e., the usual assignments) as equal as possible. Many measures for equality have been described: the range (i.e., the difference between the shortest and the longest of any customer-facility distances), the variance of these distances, the *Lorenz curve* (a tool that economists use to display income disparities), and the related *Gini*

index. It is important to realize that "equity" objectives should always be coupled with an efficiency objective, as otherwise, they tend to result in degenerate solutions. As an example, consider two customers located at the ends of a line segment, and a third customer who is just below the center of that line segment. The optimal solution for any single-facility location problem with equity objective has the facility located at the center of the circle, on whose circumference all three customers lie. This point can be very far away from the customers and, worse, as the facility moves closer all customers benefit (as all customers are now closer to the facility), but the solution is less equal. This is clearly undesirable and can only be avoided, if some efficiency objective is considered as well.

Another active area of research concerns the *location of hubs*. Hubs are an essential concept in a number of industries, most prominently the airline industry. Typically, airline flights between an origin and a destination are routed through one or two hubs. The inconvenience of having change planes once or twice is acceptable due to the fact that without hubs, many origin-destination pairs would not permit any flights between them due to low traffic volume. The flight volumes to the hubs (or concentrators) is, however, often sufficient to justify flights to them. In addition, flights between (remote) origins or destinations and more central hubs are typically done by small commuter planes, whose costs per passenger-mile are typically considerably higher than the costs of larger airplane that operate between hubs. This is the reason for inter-hub discounts, i.e., lower costs between hubs. Hub location problems are difficult to solve exactly. The major reason for the difficulties is the size of the problem. Typically, the formulation will use binary variables of the type $y_{ik\ell j}$, which assume a value of one, if the traffic from origin i to destination j is routed through hubs k and ℓ. For example, in a network with one hundred nodes, each of which is an origin, a destination, and a potential hub, there would be hundred million zero-one variables. Even considering today's powerful computing equipment, this is a very large problem.

Competitive location models were introduced by Hotelling in 1929. A mathematical statistician by trade, he considered the simplest competitive location problem one can think of: two profit-maximizing duopolists locate one facility each on a line segment. They offer a homogeneous good and compete in locations and prices. Hotelling concluded that a situation, in which the duopolists would cluster together at the center of the market, is stable. It took fifty years to prove him wrong, and it is known today that there exists no stable solution in his original version of the problem. Today, we consider two versions of competitive location models. In the first class of problems, the main quest is to find, as Hotelling did, stable solutions. These are called *Nash equilibria*. A Nash equilibrium is said to exist, if none of the participants in the game has an incentive to unilaterally move out of the current situation, which in this case means to change his location or price. The second class of problem concerns *von Stackelberg solutions*. The

economist von Stackelberg considered two groups of participants: leaders and followers. The leader will choose his actions (here: his location) first, knowing that a follower will locate later. Note that the leader's planning will require assumptions concerning the follower's objectives and perception. This is typically summarized in a reaction function, which delineates the follower's reaction to each of the leader's courses of action. In contrast, once the leader has made his decision, the follower only has to observe what the leader has done, and make his own decisions accordingly. It is apparent that the follower's problem is a conditional optimization problem (finding an optional location for the follower's facilities, given the leader's facility locations), while the leader's problem is very difficult, as even the determination of the reaction function requires the solution of a zero-one integer programming problem for each of the leader's possible courses of action.

Finally, consider *extensive facilities* and facility *layout problems*. In both areas, the facilities can no longer be represented as points on a map, so that the sizes of the facilities are no longer negligible in relation to the space they are located in. Problems of this nature are much more difficult than "simple" location problem. The main reason is that the shape of the facilities must now also be considered. Typical example of layout problems are the arrangements of work stations in an office, the placement of rooms in a hospital (operating rooms, supply rooms, recovery rooms, etc.), and the allocation of spaces in a shopping mall to stores. The best-known facility layout model is the quadratic assignment problem. Generally speaking, the purpose of this problem is to assign items to empty slots. Depending on the specific application, this may mean work functions to stations, offices to empty rooms, or drill bits to slots on a drill. One way to formulate quadratic assignment problems is to define binary variables $y_{ijk\ell}$, which assume a value of one, if item i is assigned to slot j and item k is assigned to slot ℓ, and zero otherwise. The advantage of this formulation is its linear objective function, while the disadvantage is the very large number of variables. Another formulation uses double-subscripted binary variables y_{ij}, which equal one, if item i is assigned to slot j, and zero otherwise. This formulation has much fewer variables, but its disadvantage is its quadratic objective function (hence the name of the formulation). To this day, exact solutions for quadratic assignment problems with more than about thirty items and slots remain elusive.

Exercises

Problem 1 (a location set covering problem): An administrative district includes 18 small villages. One of the functions of the district officer is to ensure that each community is reasonably well served in case of a fire. It was established that no village should be farther than 8 minutes from its closest fire hall. The graph with the villages and the distances between them is shown in the Figure 6.7. All villages must be covered.

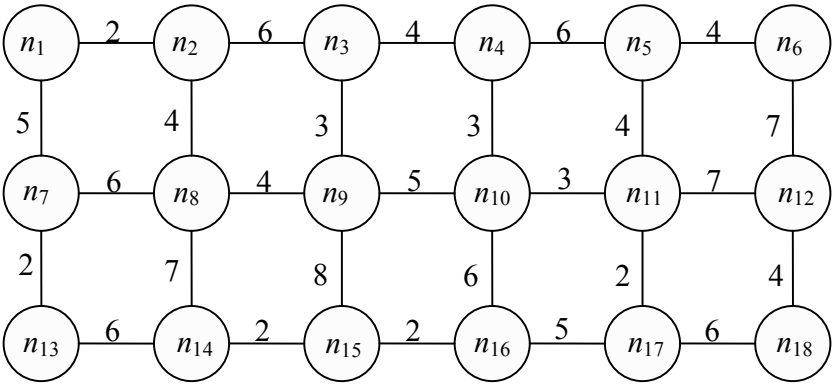


Figure 6.7

Use the reduction algorithm for the location set covering problem to reduce the problem as much as possible. First eliminate unit rows, then dominated columns, then dominated rows, then repeat as often as possible. Is it possible to obtain a solution with just the reduction rules? If so, where should the facilities be located and how many facilities are required to cover all customers? If not, try to eyeball solutions in the remaining, smaller, system and put them together with locations determined in the reduction process. How many facilities will be needed to cover all customers? The covering matrix is then shown in Table 6.2.

Table 6.2: Covering matrix for Problem 1

	n_1	n_2	n_3	n_4	n_5	n_6	n_7	n_8	n_9	n_{10}	n_{11}	n_{12}	n_{13}	n_{14}	n_{15}	n_{16}	n_{17}	n_{18}
n_1	1	1	1	0	0	0	1	1	0	0	0	0	1	0	0	0	0	0
n_2	1	1	1	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0
n_3	1	1	1	1	0	0	0	1	1	1	0	0	0	0	0	0	0	0
n_4	0	0	1	1	1	0	0	0	1	1	1	0	0	0	0	0	0	0
n_5	0	0	0	1	1	1	0	0	0	0	1	0	0	0	0	0	1	0
n_6	0	0	0	0	1	1	0	0	0	0	1	1	0	0	0	0	0	0
n_7	1	1	0	0	0	0	1	1	0	0	0	0	1	1	0	0	0	0
n_8	1	1	1	0	0	0	1	1	1	0	0	0	1	1	0	0	0	0
n_9	0	1	1	1	0	0	0	1	1	1	1	0	0	0	1	0	0	0
n_{10}	0	0	1	1	1	0	0	0	1	1	1	0	0	0	1	1	1	0
n_{11}	0	0	0	1	1	1	0	0	1	1	1	1	0	0	0	1	1	1
n_{12}	0	0	0	0	0	1	0	0	0	0	1	1	0	0	0	0	0	1
n_{13}	1	0	0	0	0	0	1	1	0	0	0	0	1	1	1	0	0	0
n_{14}	0	0	0	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0
n_{15}	0	0	0	0	0	0	0	0	1	1	0	0	1	1	1	1	1	0
n_{16}	0	0	0	0	0	0	0	0	0	1	1	0	0	1	1	1	1	0
n_{17}	0	0	0	0	1	0	0	0	0	1	1	0	0	0	1	1	1	1
n_{18}	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1	1

Solution:
Repeatedly applying the three reduction rules results in one facility located at node n_{11}. Applying the reduction rules again leaves us with the irreducible matrix:

	n_8	n_9	n_{13}
n_1	1	0	1
n_3	1	1	0
n_{15}	0	1	1

Locate facilities at 2 of the 3 columns, i.e. the solutions are (8, 9, 11) or (8, 11, 13) or (9, 11, 13). In all cases, 3 facilities are sufficient to cover *all* customers.

Problem 2 (a maximal covering location problem): A smaller rural district has the task of locating health posts in the country to serve remote villages. A village is said to be covered, if it is within ten miles of the health post. The district administration can pay for no more than two health posts. The geographical layout of the villages is shown in Figure 6.8, in which the single-digit numbers indicate the distances between the villages, and the double-digit numbers are the populations of the villages:

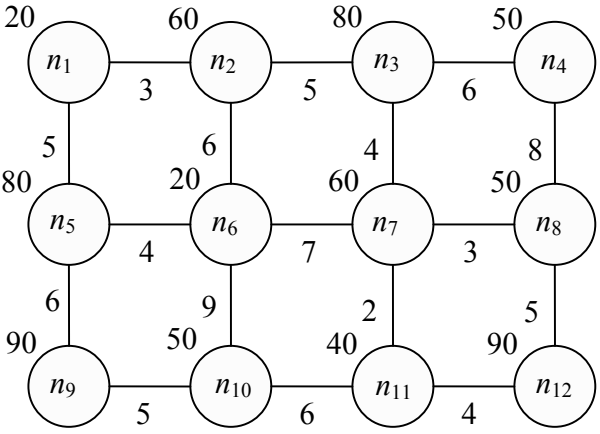


Figure 6.8

The covering matrix is then as follows:

	n_1	n_2	n_3	n_4	n_5	n_6	n_7	n_8	n_9	n_{10}	n_{11}	n_{12}
n_1	1	1	1	0	1	1	0	0	0	0	0	0
n_2	1	1	1	0	1	1	1	0	0	0	0	0
n_3	1	1	1	1	0	0	1	1	0	0	1	0
n_4	0	0	1	1	0	0	1	1	0	0	0	0
n_5	1	1	0	0	1	1	0	0	1	0	0	0
n_6	1	1	0	0	1	1	1	1	1	1	1	0
n_7	0	1	1	1	0	1	1	1	0	1	1	1
n_8	0	0	1	1	0	1	1	1	0	0	1	1
n_9	0	0	0	0	1	1	0	0	1	1	0	0
n_{10}	0	0	0	0	0	1	1	0	1	1	1	1
n_{11}	0	0	1	0	0	1	1	1	0	1	1	1
n_{12}	0	0	1	0	0	0	1	1	0	1	1	1

(a) Use the Greedy heuristic to locate the two health posts, so as to maximize the benefit of the health posts to the people.

(b) Demonstrate the Swap technique by exchanging the facility that was located first with two other facilities, one at a time. (Choose the facilities with the smallest subscripts). What are the new coverages and would you make either of the swaps permanent?

Solution: (a)

n_1	n_2	n_3	n_4	n_5	n_6	n_7	n_8	n_9	n_{10}	n_{11}	n_{12}
260	320	450	240	270	470	500	390	240	330	390	290
						↑ **max**					

n_1	n_2	n_3	n_4	n_5	n_6	n_7	n_8	n_9	n_{10}	n_{11}	n_{12}
100	100	20	0	190	190	–	0	170	90	0	0
				↑ **max**	↑ **max**						

Therefore, locate facilities at n_5 and n_7 or n_6 and n_7. Total coverage is 690.

(b) Temporarily swap n_7 and n_1. The new solution has then facilities at n_1, n_5 (or n_1, n_6) and it will cover 350 (or 550) customers, so that there will be no permanent swap.

Temporarily swap n_7 and n_2, and the new solution has facilities at n_2 and n_5 (or n_2 and n_6). It covers 410 (or 550) customers, so again, there will be no permanent swap.

Problem 3 (1-median and 1-center on a network): Industrial customers have contracted demands for heat pumps. These units are to be delivered from the warehouse of a central supplier to the companies. The supplier is now attempting to locate the warehouse, so as to minimize the transportation cost of the pumps to

its customers. The demand is fairly constant throughout the year. The delivery is per pickup truck, one heat pump at a time, resulting in a linear cost function. Figure 6.9 shows the supplier's customers, their double-digit demands, and the single-digit distances between the customers.

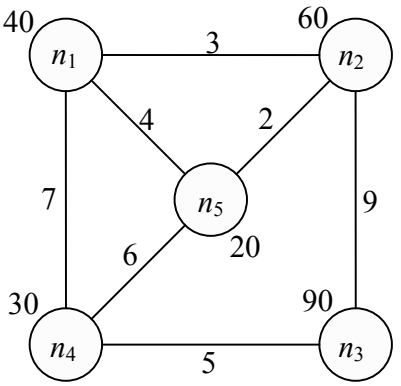


Figure 6.9

A consultant of the supplier had suggested to locate the warehouse between the nodes n_2 and n_3 at a distance of 5.4 from n_2. They have based their argument on the large weights of the adjacent nodes n_2 and n_3 that provide a strong pull to locate the warehouse there.

(a) Without any calculations, do you agree with the consultant's recommendation?
(b) Find a location on the network that minimizes the total delivery cost. How much more expensive was the consultant's recommendation?
(c) Ignore now the weights at the nodes, and assume that the same graph were to be used by some planner to locate a vertex 1-center. Where would this center be located?

Solution: (a) No. This is a 1-median problem and there is more to a solution than just weight.

(b) $\mathbf{w} = [40, 60, 90, 30, 20]$, $\mathbf{D} = \begin{bmatrix} 0 & 3 & 12 & 7 & 4 \\ 3 & 0 & 9 & 8 & 2 \\ 12 & 9 & 0 & 5 & 11 \\ 7 & 8 & 5 & 0 & 6 \\ 4 & 2 & 11 & 6 & 0 \end{bmatrix}$

Then $\mathbf{wD}$ = [1,550; 1,210; 1,390; 1,330; 1,450], so that node n_2 is the optimal location and the total transportation costs are 1,210. The consultant's

recommendation is [40, 60, 90, 30, 20] [8.4, 5.4, 3.6, 8.6, 7.4] = 1,390. The costs are about 15% higher than optimal.

(c) The question is to determine a 1-node center. The column maxima are [12, 9, 12, 8, 11] with 8 being the minimum, so that node 4 will be optimal.

Problem 4 (1-node center in the plane with Manhattan distances): A rural district is planning the location of its fire station. The station will have to serve seven villages. The coordinates of the villages are (0, 0), (6, 1), (2, 3), (8, 3), (0, 4), (4, 5), and (3, 7). As the people in the district are fierce defenders of property rights, the roads were constructed parallel to the rectangular fields.

(a) What is the optimal minimax location of the fire station?

(b) Suppose that it takes the fire truck and its crew 3 minutes to get ready after an emergency call, 2 minutes to drive one distance unit, and another 3 minutes to get the hoses and pumps going. Given the solution determined under(a), how much time elapses between the phone call and the beginning of the fire fighting action at the site of the most distant customer? How long does it take on average?

Solution:

(a) $\alpha_1 = \max\{0, 7, 5, 11, 4, 9, 10\} = 11$,
$\alpha_2 = \max\{0, -5, 1, -5, 4, 1, 4\} = 4$,
$\alpha_3 = \max\{0, 5, -1, 5, -4, -1, -4\} = 5$, and
$\alpha_4 = \max\{0, -7, -5, -11, -4, -9, -10\} = 0$, so that

$\alpha_5 = \max\{11, 9\} = 11$.

Hence the optimal coordinates of the facility are
$x = \frac{1}{2}(5) = 2\frac{1}{2}$ and $y = \frac{1}{2}(6) = 3$, as well as
$x = \frac{1}{2}(7) = 3\frac{1}{2}$ and $y = \frac{1}{2}(4) = 2$,
both with $z = \frac{1}{2}(11) = 5\frac{1}{2}$.

(b) Worst case: $6 + 2(5\frac{1}{2}) = 17$ minutes. Average case: $6 + (1/7)57 = 14.14$ min.

Problem 5 (*p*-node center in a graph): Consider again Figure 6.9 and ignore the weights at the nodes. For $p = 2$, determine the vertex p-center with the bisection search algorithm.

Solution:

$\overline{D} = (n-1)\max\{d_{ij}\} = 4(9) = 36$, $\underline{D} = 0$.

$D = 18$. Set cover: $p(D) = 1$, locate anywhere. $\overline{D} = 18$.

$D = 9$. Set cover: $p(D) = 1$ with facility at node n_2. $\overline{D} = 9$.

$D = 4$. Set cover: $p(D) = 3$ with facilities at nodes 4, 3, and one of 1, 2, 5. $\underline{D} = 5$.

$D = 7$. Set cover: $p(D) = 2$, with facilities at nodes 1 and 3. $\overline{D} = 7$.

$D = 6$. Set cover: $p(D) = 2$, with facilities at nodes 1 and either 3 or 4. $\overline{D} = 6$.

$D = 5$. Set cover: $p(D) = 2$ with facilities at 1 and either 3 or 4. $\overline{D} = 5$.

Now $\overline{D} = 5 = \underline{D}$, Stop.

Problem 6 (1-median problem in the plane with Manhattan distances): Customers are located at (0, 0), (7, 0), (6, 3), (5, 2), (5, 7), (3, 4), (6, 6), and (2, 8) with weights 20, 40, 30, 50, 10, 60, 70, and 80.

Given Manhattan distances, determine the point that minimizes the sum of distances to a single new facility. What are the total weighted costs from the new facility to the customers?

Solution: The sum of weights equals 360, so that the location of the facility should be at (5, 4), with distances to the customers of 9, 6, 2, 2, 3, 2, 3, and 7, so that the weighted sum (total costs) equals 1,500.

Problem 7 (center-of-gravity and Weiszfeld method): Customers are located at (0, 0), (6, 0), and (10, 5) with weights 4, 7, and 2, respectively.

(a) Calculate the center-of-gravity, i.e., the optimal solution of the minisum location problem with squared Euclidean distances.
(b) Start with an initial guess of (1, 1) and perform one iteration with Weiszfeld's method to determine the 1-median.

Solution: (a) center-of-gravity is at (62/13, 10/13) ≈ (4.77, 0.77).

(b)

$$x = \frac{\frac{0}{\sqrt{2}} + \frac{42}{\sqrt{26}} + \frac{20}{\sqrt{97}}}{\frac{4}{\sqrt{2}} + \frac{7}{\sqrt{26}} + \frac{2}{\sqrt{97}}} = \frac{10.2676}{4.4043} = 2.3313\,, \quad y = \frac{\frac{0}{\sqrt{2}} + \frac{0}{\sqrt{26}} + \frac{10}{\sqrt{97}}}{\frac{4}{\sqrt{2}} + \frac{7}{\sqrt{26}} + \frac{2}{\sqrt{97}}} = \frac{1.0153}{4.4043} = 0.2305\,.$$

Problem 8 (*p*-median with Greedy and vertex substitution heuristics): Consider the undirected graph shown in Figure 6.10:

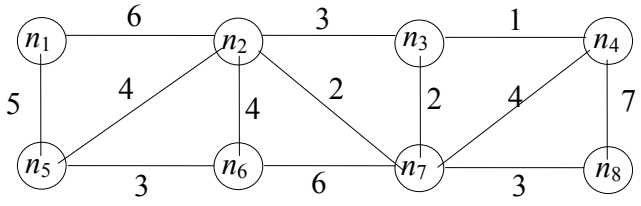


Figure 6.10

Assume that the customers n_1, n_2, ..., n_8 have demands of 38, 25, 13, 18, 15, 21, 32, and 40, respectively. Suppose now that the coverage distance is $D = 4$.

(a) Set up the capture table and apply the Greedy heuristic to locate 3 facilities. Where should the facilities be located and what is the total capture?
(b) Use the vertex substitution heuristic to improve the solution.

Solution:

(a)

Potential facility location	n_1	n_2	n_3	n_4	n_5	n_6	n_7	n_8
Capture	38	124	88	88	61	61	128	72

Locate one facility at n_7.

Potential facility location	n_1	n_2	n_3	n_4	n_5	n_6	n_8
Capture	38	36	0	0	36	36	0

Locate one facility at n_1.

Potential facility location	n_2	n_3	n_4	n_5	n_6	n_8
Capture	36	0	0	36	36	0

Locate one facility at either n_2, n_5 or at n_6. Hence the solution locates facilities at either (n_7, n_1, n_5) or at (n_7, n_1, n_6) with a total capture of 202. The facilities capture everything, so there can be no better solution.

(b) Vertex substitution procedure: For example, start with facilities located at (n_7, n_1, n_5). Exchange n_1 and n_2. The exchange leads to facilities located at (n_7, n_2, n_5) that has a capture of 164 as opposed to the previous solution that has 202. Do not swap.

Problem 9 (*p*-median in a network with Greedy, location-allocation heuristic): Consider the graph in Figure 6.11.

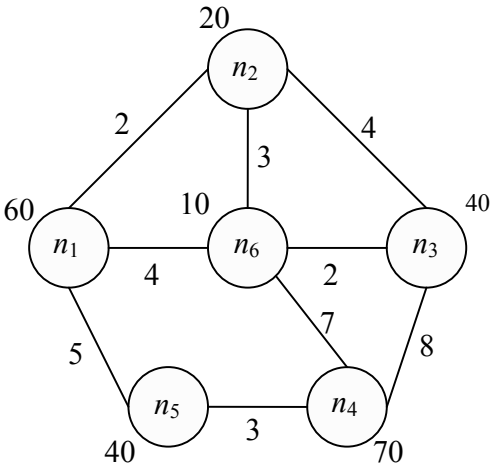


Figure 6.11

(a) Determine the 2-median by using the Greedy heuristic.
(b) Improve the solution found under (a) by the location-allocation heuristic.
(c) Ignore the solution found in (b) and improve the solution in (a) by the vertex substitution method.

Solution:

(a) The weights are **w** = [60, 20, 40, 70, 40, 10] and the distance matrix is

$$\mathbf{D} = \begin{bmatrix} 0 & 2 & 6 & 8 & 5 & 4 \\ 2 & 0 & 4 & 10 & 7 & 3 \\ 6 & 4 & 0 & 8 & 11 & 2 \\ 8 & 10 & 8 & 0 & 3 & 7 \\ 5 & 7 & 11 & 3 & 0 & 9 \\ 4 & 3 & 2 & 7 & 9 & 0 \end{bmatrix}.$$

Locate the first facility: **wD** = [1,080, 1,290, 1,460, 1,190, 1,180, 1,230], so that the first facility is located at n_1.

For the second facility, we obtain costs of [–, 950, 820, 440, 530, 810], so that the second facility will be located at n_4 for a total z-value of 440.

(b) Allocation: To the facility at n_1, we allocated customers at nodes n_1, n_2, n_3, and n_6, and to the facility at n_4, we allocate the customers at n_4 and n_5. The optimization on the two problems then results in costs vectors [320, 310, 460, 380] with the minimum occurring at n_2, and [120, 210] with the minimum at n_4. The next step then allocates to the facility at n_2 the customers at n_1, n_2, n_3, and n_5, to the facility at n_4, the customers at n_4 and n_5, which is the same as before. Hence the method has converged with an optimal solution with facilities at n_2 and n_4 with a total cost of 310 + 120 = 430.

7 Project Networks

Back in the days when projects were dealt with by a single individual or a group of workers working sequentially, there was no need for project networks. As an example, consider the construction of a house. Somewhat simplistically, assume that a single individual wants to build a log cabin. He will first dig a hole in the ground for the foundation, then pour the cement, then lay the logs one by one, and so forth. Each job is completely finished before the next task begins. This is a sequential plan, and there is very little that can be done as far as planning is concerned. Consider, however, some of the issues that have arisen as a result of the division of labor. Nowadays, the plumbers can work at the same time the electrician does, but not before the walls have been established, which is also required for the roof to be put up. Given these interdependencies, planning is necessary in case time is an issue. Clearly, while it is *possible* that, say, electrician and plumber can work in the building at the same time, it is *not necessary* to use this parallelism: we can still have the two contractors work one after the other, if we so wish. The project will take longer, but it is possible.

Project networks were designed by a number of firms in the 1950s. The so-called c*ritical path method* (or *CPM* for short) was developed by du Pont de Nemours and the Remington Rand Univac corporations for construction projects. At roughly the same time, the United States Navy in conjunction with the Lockheed Aircraft Corporation and the consulting firm of Booz, Allen, and Hamilton devised the *Program Evaluation and Review Technique* (*PERT*) for their Polaris missile program. Even though *CPM* and *PERT* have completely independent backgrounds, today we can consider them very close brothers: their underlying ideas are identical, the resulting networks are identical, and the only difference is that *CPM* is a deterministic technique, *PERT* is (partially) stochastic.

This chapter is organized as follows. The first section will introduce the elements of the critical path method, demonstrate its graphical representation, and describe basic planning with the critical path method. The remaining sections of this chapter deal with extensions of the basic concept: the second section allows the acceleration of the project (a process that introduces costs into the model), the third section allows resources to be used (which, with the obvious limitations,

H.A. Eiselt and C.-L. Sandblom, *Operations Research: A Model-Based Approach*,
DOI 10.1007/978-3-642-10326-1_7, © Springer-Verlag Berlin Heidelberg 2010

results in an optimization model that allocates the scarce resources), and the final section of this chapter discusses the probabilistic *PERT* method.

7.1 The Critical Path Method

Before getting in to details, the planner will have to decide on what level the planning will take place. We may be interested in planning on the macro level in order to get the "bigger picture." As an example, when building a house, we may have "foundation work" as a single task, another one is "electrical work," another is "plumbing," another is "roofing," and so forth. Zooming in closer, we may look at each such task as a project in itself. For instance, plumbing may include subtasks such as the installation of pipes, connections to outside lines or septic systems, etc. Zooming in even further, the installation of lavatories may be further subdivided into the mountain of the washbasin, the connection of the faucet to the pipes, and so forth. We can construct project networks on each of these levels.

Once the level has been decided upon, the task in its entirety—the building of the house, design and planting of a public or private garden, an individual's University studies—will be referred to as a *project*. Each project can now be subdivided into individual *tasks* or *activities*. Planting a bed of grape hyacinths, installing the flashing on the roof, or taking a specific course in a University program are typical examples of such activities. Associated with each activity is a *duration* that indicates how long it takes to complete the activity. In *CPM*, the durations of all activities are assumed to be known with certainty (which is the only distinction to *PERT*, where the durations have underlying probability distributions).

In addition to the activities and their durations, we also need to have *precedence relations*. Such relations indicate which activities must be completely finished before another activity can take place. For example, in order to start the activity "drive with the car to grandmother," the activities "gas up the car," "pack the gifts (wine and cheese) for grandmother," and "lock the house" must be completely finished. A complete set of precedence relations will specify the (immediate) predecessors of each activity.

In the analysis in this section, we are only concerned about time. There is no optimization that takes place here, all we want to know what the earliest time is by which the project can be completed. And while making these calculations, we can also find out when each task can and must be started and finished. That allows us to determine bottleneck activities in the project, whose delay will delay the entire project. In subsequent sections, we will include other components in the basic network, including money and other resources.

One of the huge advantages of project networks and one of the reasons for their popularity among users is the ease with which they can be understood and

visualized. Following the old adage that "a manager would much rather live with a problem he cannot solve than with a solution he cannot understand," many managers have adopted project networks as a standard tool in their toolkit. This was helped tremendously be a change in the representation that was made some time during the 1990s. Traditionally, each activity was represented by an arc in a network, while nodes represented events (which are no more than the beginning and/or the end of activities). Such a representation is referred to as an *activity-on-arc* (*AOA*) *representation*. While *AOA* networks have some obvious advantages, they are rather difficult to construct and may require a lot of artificial activities (called *dummy activities*). A more modern way to visualize relations between activities is the *activity-on-node* (*AON*) *representation*. In it, each node represents an activity, while the arcs represent the precedence relations. In addition to the given activities, each project network contains two artificial nodes n_s and n_t, which are the starting node (the *source*) and the terminal node (the *sink*), which symbolize the beginning and the end of the project. The project start has arcs leading out of it to all activities that do not have any predecessors, while all nodes that represent activities without successors will have arcs leading directly to the terminal node "Project End." For all other activities j, we introduce an arc from node i to node j, if activity i is an immediate predecessor of j.

As a numerical illustration, consider the precedence relations shown in Table 7.1.

Table 7.1: Precedence relations for sample network

Activity	Immediate predecessor	Duration (in weeks)
A	–	5
B	*A*	3
C	*A*, *B*	7
D	*B*	4
E	*B*, *C*	6
F	*C*, *D*, *E*	4
G	*D*	2
H	*F*, *G*	9
I	*F*, *G*	6
J	*I*	2

The network that includes all of the activities and precedence relations is shown in Figure 7.1. Like each project network, it has a unique start n_s, a unique end n_t, all arcs are directed, and no cycles can possibly exist, as they would require an activity to be completely finished before it can actually start, which is obviously impossible.

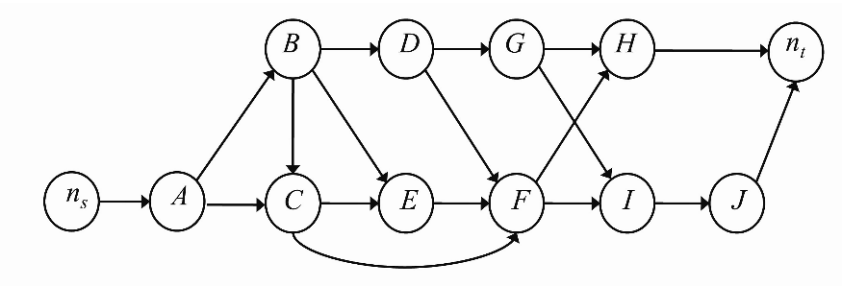


Figure 7.1

As an example for (nested) precedence relations, consider activity *F*. It is clear from the project network that the activities *C*, *D*, and *E* must be completely finished before activity *F* can commence. However, activity *C* requires that *A* and *B* are completely finished, activity *D* requires that *B* is finished, and activity *E* requires that *B* and *C* are completely finished. So, in reality activity *F* cannot commence before the activities *A*, *B*, *C*, *D*, and *E* are all finished. This does not present a difficulty, but users should be aware of this. Incidentally, it is not necessary to specify that activity *C* is a direct predecessor of *F* since *E* must precede *F* anyway, and *C* precedes *E*.

We are now interested in finding the earliest possible time at which the entire project can be completed. For computational convenience, we assume that the project start occurs at time $t = 0$ and the end of the project occurs at time *T*, which will be determined in the process. Consider now one of the many (14, to be precise) paths of the network that lead from n_s to n_t, say $n_s - A - C - E - F - H - n_t$. Starting at the end and moving back towards the beginning one step at a time, we notice that n_t can only finish if *H* is completely finished, which, in turn, requires that *F* is completely done, which, in turn, requires that *E* is fully done, etc. The length of any path from source to sink is defined as the sum of durations of all of its activities. In this example, the path we are presently looking at has the length of 5 + 7 + 6 + 4 + 9 = 31. This also means that while there are other precedence relations, the ones on this path result in a project duration of 31, which is thus a lower bound on the project duration. From this, we can conclude that the project duration is equal to the length of the longest path in the network.

While it would certainly be possible to determine the longest path in a network by enumeration (even though there will be a lot of paths), a different type of computation is preferred as it will provide decision makers with additional and very valuable information. This procedure will be described in the following.

The first part of the procedure uses what is commonly known as a *forward pass*, a *forward sweep*, or a *forward recursion*. In the forward recursion, we compute the *earliest possible starting times* (*ES*) of all activities, as well as their *earliest possible finishing times* (*EF*). We start labeling the nodes in the forward pass with the source node. The earliest starting time of the source node is arbitrarily set to

$ES(n_s) = 0$. For any node the earliest possible finishing time *EF* equals the earliest possible starting time *ES* plus the duration of the activity *t*. As a result, $EF(n_s) = 0$. In order to continue labeling (which in the forward pass means assigning a value *ES* to a node), we will use the following

Rule 1: In the forward pass, a node can only be labeled if all of its predecessors have been labeled.

If more than one node satisfies the condition in Rule 1, labeling continues with any of the nodes that satisfy the condition. The label *ES* of a node is then the maximum among *all EF* labels of its direct predecessors. The reason for this calculation is easily explained. Suppose that an activity has three direct predecessors with earliest possible finishing times of 4, 7, and 9, respectively. It is apparent that we cannot start before 4, as none of the predecessors has been finished. Between time 4 and 7, the first predecessor is finished, but we will have to wait until all predecessors have been finished, which is the case at time 9.

Applying this rule to our example means that, once n_s has been labeled, we can now label node *A*. Node *A* has only one predecessor, so that $ES(A) = EF(n_s)$. Given that, we can calculate $EF(A) = ES(A) + t_A = 0 + 5 = 5$. Now that node *A* is labeled, we can continue labeling with nodes *B* and *C*. Arbitrarily choose node *B*. Activity *B* also has only one predecessor, which is *A*. Consequently, $ES(B) = EF(A) = 5$, and $EF(B) = ES(B) + t_B = 5 + 3 = 8$. Now that activity *B* has been labeled, we can continue labeling node *C*. As activity *C* has activities *A* and *B* as predecessors, we have $ES(C) = \max\{(EF(A), EF(B)\} = \max\{5, 8\} = 8$, so that $EF(C) = ES(C) + t_C = 8 + 7 = 15$. Labeling can now continue with nodes *D* and *E*. The results of the forward labeling phase are summarized in Table 7.2.

Table 7.2: The earliest possible starting and finishing times

Activity	n_s	*A*	*B*	*C*	*D*	*E*	*F*	*G*	*H*	*I*	*J*	n_t
Earliest possible starting time *ES*	0	0	5	8	8	15	21	12	25	25	31	34
Earliest possible finishing time *EF*	0	5	8	15	12	21	25	14	34	31	33	34

At this point, the sink n_t has been labeled, and the *ES* and *EF* labels of the sink (which are necessary equal, as the artificial activity n_t has zero duration), indicate the total duration of the project. In other words, we now know that the project can be completed in $T = 34$ weeks. This terminates the forward sweep.

In the backward sweep, we will compute the *latest allowable finishing times LF* and the *latest allowable starting times LS* of all activities (in that order). By "latest allowable," we refer to the time we can start or finish an activity, given that the project has to be completed by time *T* (in our example $T = 34$). As the forward

sweep started with the source being labeled first, the backward sweep commences by labeling the sink. Now we consider a node labeled, if we have computed its *LF* and *LS* values. As the project duration *T* is now known, we will label the sink node $LF(n_t) = LS(n_t) = T$. The rule for labeling other nodes is now

Rule 2: In the backward pass, a node can only be labeled if all of its successors have been labeled.

With the sink as the only labeled node, only the activities *H* and *J* can be labeled. Arbitrarily choose node *H*. This node has only one successor, so that $LF(H) = LS(n_t) = 34$. Given that activity H must be finished no later than 34 and its duration is $t_H = 9$, we determine that $LS(H) = LF(H) - t_H = 34 - 9 = 25$. The process for node *J* is similar, and we obtain $LF(J) = LS(n_t) = 34$, and $LS(J) = LF(J) - t_J = 34 - 2 = 32$. At this point, the nodes n_t, *H* and *J* are labeled, so that we can continue labeling only with node *I* (as node *G* has node *I* as a successor which is not yet labeled). Node I has again only one successor, so that its label can easily be calculated as $LF(I) = LS(J) = 32$ and $LS(I) = LF(I) - t_I = 32 - 6 = 26$. At this point, we can continue labeling the nodes *G* and *F*. Arbitrarily choose *F*. Node *F* has activities *H* and *I* as successors. In order to determine its latest finishing time, consider this. As we have just computed, its two successors *H* and *I* cannot start any later than 25 and 26, respectively. If activity *F* were to finish any later than, say, 25½, activity *H* could not start on time. In other words, in order to avoid delaying the entire project it is necessary that *all* activities can start on time, so that $LF(F) = \min\{LS(H), LS(I)\} = \min\{25, 26\} = 25$. The resulting latest allowable starting time is then $LS(F) = LF(F) - t_F = 25 - 4 = 21$. The next node to be labeled is *G*. Since its successors are *I* and *H*, we have $LF(G) = \min\{LS(I), LS(H)\} = \min\{26, 25\} = 25$, and $LS(G) = LF(G) - t_G = 25 - 2 = 23$. The procedure continues in this fashion until the source is labeled. The labels of all nodes are shown in Table 7.3.

Table 7.3: The latest allowable starting and finishing times

Activity	n_s	*A*	*B*	*C*	*D*	*E*	*F*	*G*	*H*	*I*	*J*	n_t
Latest allowable starting time *LS*	0	0	5	8	17	15	21	23	25	26	32	34
Latest allowable finishing time *LF*	0	5	8	15	21	21	25	25	34	32	34	34

A good test for correctness is to examine $LF(n_s)$ in the backward sweep, this value must be zero. If it is not, an error has been made. However, the converse is not true: even if $LF(n_s) = 0$, it does not mean that the backward recursion has been done correctly.

Now that all earliest possible and latest allowable starting and finishing times have been computed, we are able to determine which of the activities are critical when

scheduled and which are not. As an example, consider activity G. So far, we have determined that the earliest possible time that we can schedule the activity is at $ES(G) = 12$, while the latest possible finishing time of G is $LF(G) = 25$. This gives us a time window of $25 - 12 = 13$ weeks during which the activity has to be scheduled. That is not problem, since the duration of the activity is only 2 weeks, so that there is plenty of leeway. An expression of the magnitude of this leeway is the total float of the activity G, which we will abbreviate here as $TF(G)$. The total float of an activity is the magnitude of the time window for its schedule minus the duration of the activity. Formally for activity G, we have $TF(G) = LF(G) - ES(G) - t_G = 25 - 12 - 2 = 11$. Decision makers can use the information provided by the total float as the amount of time by which the duration of an activity can increase without delaying the entire project. Thus, activities with a large float are safe in that their duration can increase significantly without delaying the entire project. On the other hand, an activity with a large float indicates that there are too many resources allocated to this activity. Redirecting these resources elsewhere may result in possible decreases of the durations of other activities.

As another example, consider activity H. Its time window ranges from $ES(H) = 25$ to $LF(H) = 34$. With its duration of $t_H = 9$, we can compute the activity's total float as $TF(H) = LF(H) - ES(H) - t_H = 34 - 25 - 9 = 0$. This indicates that there is absolutely no leeway when scheduling activity H. Activities that have no leeway in the schedule are referred to as *critical activities*. A critical activity has the property that as soon as its duration increases, the project will be delayed, regardless how small the time increase actually is.

Critical activities are very similar to those resources in optimization problems that are satisfied as equations at optimum and thus represent bottlenecks in the problem. On the other hand, noncritical activities in project networks can be compared to resources or constraints that have positive slacks or excess variables at optimum. Note that for the ease of computations, we can calculate the total float also as $TF = LF - EF = LS - ES$.

In order to have all information available at a glance, it is useful to draw the network in a slightly different way. Rather than representing each node by a circle with its name in it, we suggest to represent a node by a box with nine different fields as shown in Figure 7.2. We will refer to the different fields by the geographic directions they are found in, e.g., the field in the North, the Southwest, etc. The center of the node is reserved for the name of the activity (here activity D). The fields in the North and in the South both show the duration t_D of the activity.

Recall that during the forward sweep, we compute the earliest possible starting time $ES(D)$ and the earliest possible finishing time $EF(D)$ of the activity under consideration. They are found in the Northwest and the Northeastern fields, respectively. In the backward sweep, we determine the latest allowable starting time $LS(D)$ and the latest allowable finishing time $LF(D)$ of the activity. This

information is put into the fields in the Southwest and Southeast, respectively. Finally, the field in the East will include the total float *TF*, which is computed after the forward and backward passes have been completed. The field in the West will remain empty for now. It is designed for the resource consumption of the activity, which is not considered in the basic model.

$ES(D)$	t_D	$EF(D)$
—	D	$TF(D)$
$LS(D)$	t_D	$LF(D)$

Figure 7.2

The three fields in the top row of each node read from left to right symbolize the relation $ES + t = EF$, while the three fields at the bottom of a node read from right to left show the relation $LF - t = LS$. This is why we have chosen to include the duration of the activity twice, once in the field in the North and again in the South.

Rather than using tables for the display of the information, we can work directly on the graph, which is much easier, as it provides all required information at a glance. The project network for our example is shown in Figure 7.3.

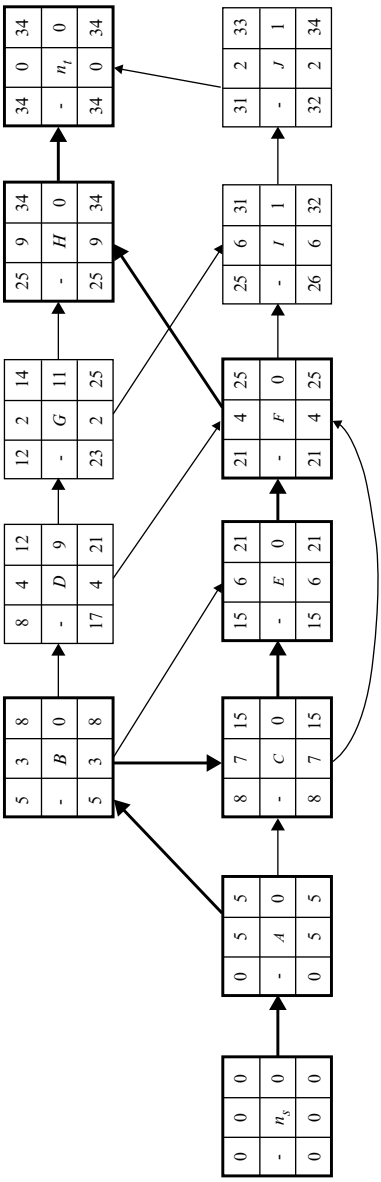

0 0 0
0 0 0
- n_S 0
0 0 0
0 5 5
0 5 5
- A 0
0 5 5
5 8 8
3 3 0
- B 3
5 5 8
8 15 15
7 7 0
- C 7
8 8 15
8 12 21
4 4 9
- D 4
17 17 21
15 21 21
6 6 0
- E 6
15 15 21
12 14 25
2 2 11
- G 2
23 23 25
21 25 25
4 4 0
- F 4
21 21 25
25 34 34
9 9 0
- H 9
25 25 34
25 31 32
6 6 1
- I 6
26 26 32
31 33 34
2 2 1
- J 2
32 32 34
34 34 34
0 0 0
- n_t 0
34 34 34

Figure 7.3

Having all of this information at hand, we are now able to determine what gave the method its name, *viz*., the *critical path*. Formally, the critical path is a path from the source n_s to the sink n_t that includes only critical activities, such that node j can directly succeed node i on the critical path, only if $ES(j) = LF(i)$. The critical path in our example is shown in bold lines and it includes the activities $n_s - A - B - C - E - F - H - n_t$. Clearly, the length of that path, obtained by adding its activity durations, equals $T = 34$. Note that the critical path does *not* include the link from node B to node E, even though E directly follows B, but $ES(E) = 15 > 8 = LF(B)$. This clearly indicates that it is not sufficient to simply connect all neighboring nodes that have zero total float. Furthermore, it may happen that a project network has multiple critical paths. The next section will have examples of that case.

We conclude this section by summarizing the procedure that determines the critical path:

(1) Use the forward pass to calculate the earliest possible starting and finishing times of all activities.
(2) Use the backward pass to calculate the latest allowable starting and finishing times of all activities.
(3) Calculate the total floats of all activities.
(4) Determine the critical path.

7.2 Project Acceleration

So far, we have considered time as the only criterion in project networks. Also note that the technique described in the previous section did not involve any optimization, all we have done is determined when the project can be finished and which of the activities are bottlenecks in the system. In this section, we will return to the basic model, but allow the possibility to accelerate individual activities, so as to be able to finish the project earlier. The result will be a list that shows possible finishing times of the project and the amounts that will have to be paid to reach them. This will enable the planner to decide what combination of money spent and project duration best fits the specific situation.

In order to describe the situation, consider a single activity. As before, the activity will have what we now call a *normal duration*. Since we will engage in a marginal analysis, the cost of the activity at its normal duration are immaterial (we will have to engage in the activity in any case), and the only costs we consider are those that are incurred due to the acceleration of the activity. Suppose that the normal duration of our activity is 7 hours. It is now possible to use more resources (e.g., more manpower, more tools, contracting out part of the activity, or any similar measure) to accelerate this activity. Suppose that it costs $20 to reduce the duration of the activity to 6 hours. Using more resources still, additional money

can reduce the duration of the activity further. For simplicity, we assume that the cost function of the acceleration is linear, meaning that reducing the duration by another hour to 5 hours costs another $20 for a total of $40. Note that normally the cost function is increasing, meaning that reducing the duration by one hour costs, say $\$x$, reducing it by another hours costs more than $\$x$, another reduction is more expensive still, and so forth.

It is quite apparent that the reduction has some limitations, below which we cannot reduce the duration of the activity any further. The shortest activity duration of an activity that can be achieved is customarily referred to as *crash time*, and the process of acceleration is sometimes called *crashing*. Our task is now to determine which activities should be accelerated or crashed, so as to achieve the desired result at the lowest possible cost.

As an illustration of the concept, consider the numerical example shown in Figure 7.4.

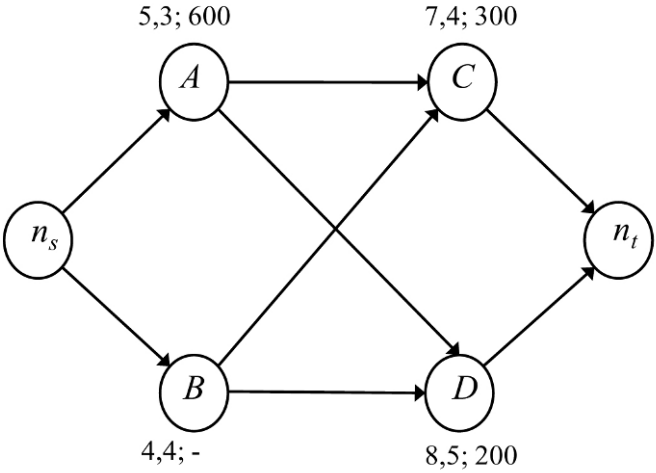


Figure 7.4

The project has four activities *A*, *B*, *C*, and *D*, whose normal times, crash times, and unit acceleration costs are shown next to the nodes. For example, activity *D* normally takes 8 hours (there are no costs incurred at this duration), but we can reduce the duration down to 7, 6, or 5 hours. Each hour of acceleration costs $200. Note that activity *B* cannot be accelerated.

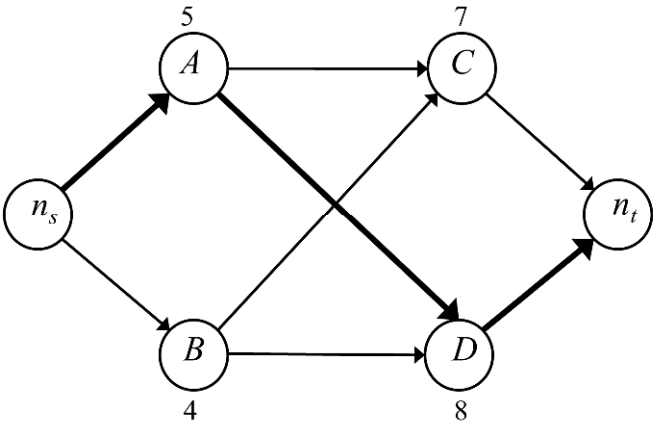


Figure 7.5

Figure 7.5 shows the project network under consideration along with the normal durations of the activities (in hours) and the critical path, which is shown in bold arcs. (Note that we normally have to use the forward sweep/backward sweep procedure in each step, but since the project network here is so small, we may enumerate the four paths *A*–*C*, *A*–*D*, *B*–*C*, and *B*–*D*, determine their respective lengths and choose the longest path; it is the critical path). The present duration of the project is 13 hours.

We now have to determine which activities to accelerate. Recall that the project duration is determined by the length of the longest path in the network. This means that as long as we are not accelerating an activity on the longest, i.e., the critical path, the project duration will not be reduced. This leads to the important realization that we must accelerate an activity on the critical path. And, among those activities, we will choose the one that minimizes our marginal, i.e., additional costs. In our example we have a choice between either accelerating activity *A* at a cost of \$600, or activity *D* at a cost of \$200. Since it is less expensive to accelerate activity *D*, we reduce its duration by a single hour to 7 hours. We now have a new network (even though the networks structure has not changed and never will during the computations, only one activity duration has changed) and we have to determine the critical path and the project duration anew. The result is shown in Figure 7.6.

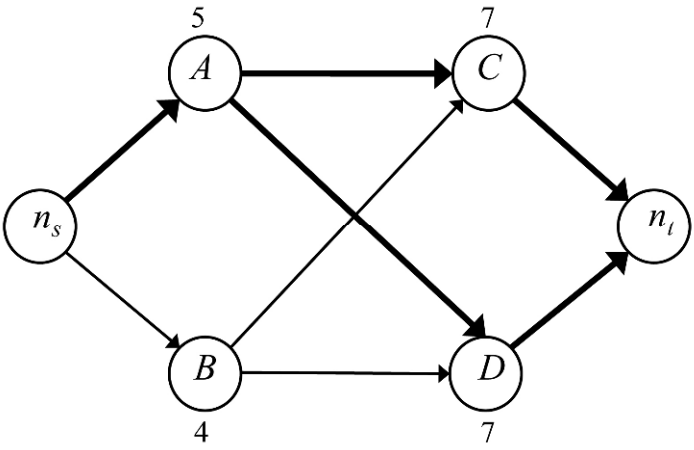


Figure 7.6

We notice that there are now two critical, i.e., longest paths in the network. They are $n_s - A - C - n_t$ and $n_s - A - D - n_t$, both having a length of 12. Accelerating the project further will pose some additional difficulties. In order to demonstrate these problems, suppose that we were to again accelerate activity D. this would cause the path $n_s - A - D - n_t$ to be only 11 hours long, while the path $n_s - A - C - n_t$ would still be 12 hours long and as such, would now be the only critical path. In other words, we would have spent another \$200 and still would have to wait 12 hours to finish the project. This means that we have to refine the rule somewhat that tells us which activities must be accelerated in order to speed up the project. In fact, we will have to accelerate a set of activities, so that at least one of the activities in this set is on each of the critical paths. For commercial uses, this can be accomplished by network techniques; for our purpose we examine the network and enumerate the possibilities. In the network in Figure 7.6, we can either accelerate activity *A* (at a cost of \$600), or the activities *C* and *D* (at a cost of \$300 + \$200 = \$500). Before making the actual decision, we have to ascertain that all of these accelerations are actually possible, i.e., that the present durations are all above the crash times. In our case, the activities *A*, *C*, and *D* have present durations of 5, 7, and 7, while their crash times are 3, 4, and 5, so that all activities can actually be accelerated. Since the cheapest option is to accelerate activities *C* and *D*, we accelerate each of these activities by one unit each. Based on the new activity durations, we also determine the new critical path(s). The results are shown in Figure 7.7.

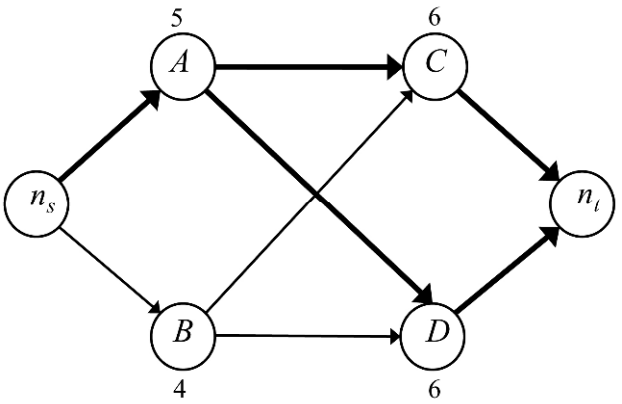


Figure 7.7

If any further accelerations is required, the options are the same as before. Checking the possibility to accelerate, few find that the present activity durations of the nodes on the critical path A, C, and D are 5, 6, and 6, while their crash times are 3, 4, and 5, so durations of all of these activities can be reduced further. The least expensive option again involves accelerating activities *C* and *D* by one hour each, resulting in the situation shown in Figure 7.8.

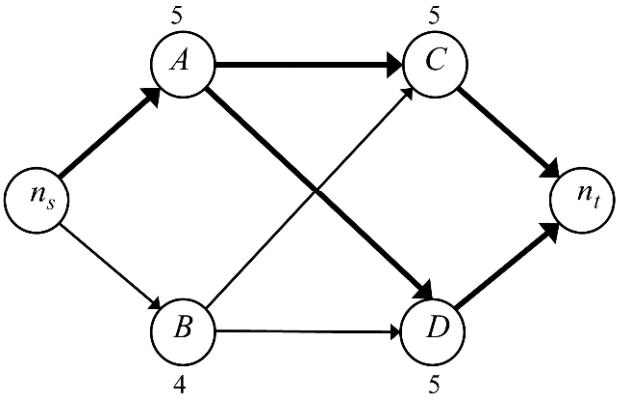


Figure 7.8

While the critical paths are still the same, the situation has changed. Activity *D* has now reached its crash time and no longer can be accelerated. This means that the only way to accelerate both critical paths simultaneously is to speed up activity *A* at a cost of $600. The resulting situation is shown in Figure 7.9.

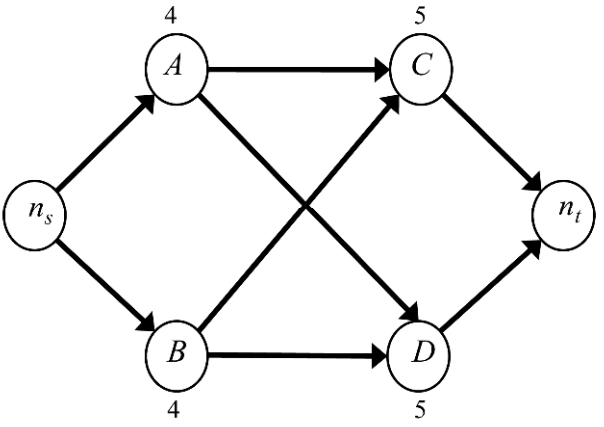


Figure 7.9

Notice that at this point in time, all four paths from the source to the sink are critical. This means that the next acceleration will have to ensure that at least one activity on each path in the network is accelerated. However, in this case at least one path, *viz*., the path $n_s - B - D - n_t$ can no longer be accelerated at all: both activities *B* and *D* are at their respective crash times. This means that the shortest possible project duration is $T = 9$ and it can be achieved at a cost of \$1,800. The results achieved in the above computations are shown in Table 7.4.

Table 7.4: Summary of the project acceleration process

Accelerate activity	Total acceleration costs	Critical path	Project duration
—	0	*A–D*	13
D	0 + 200 = 200	*A–C and A–D*	12
C and D	200 + 500 = 700	*A–C and A–D*	11
C and D	700 + 500 = 1,200	*A–C and A–D*	10
C and D	1,200 + 600 = 1,800	*A–C*, *A–D*, *B–C*, *and A–D*	9

This is where the decision maker comes in. He can now determine what it costs to accelerate the project and whether or not it is worth it. This is a good example of what operations research does best: prepare decisions (rather than actually make them).

A few concluding comments are in order. It became clear that as the process moved forward, more and more paths became critical, implying that more and more activities had to be accelerated in order to reduce the project duration further. This does, of course, imply increasing costs from each unit of acceleration. This

was to be expected, of course: at first it is easy to accelerate a project, but as the timeframe becomes tighter and tighter, the costs skyrocket.

If it appears too cumbersome to go through the entire process, a quick idea of how short the project duration can actually be is to crash all activities and determine the critical path on that basis. While this will certainly result in the shortest possible project duration, it is usually not necessary to crash all activities to reach the same overall duration. A good illustration is the above example. The optimized durations of 4, 5, 4, and 5 of the activities *A*, *B*, *C*, and *D* were sufficient to reduce the project duration to 9, while the crash durations of the activities are 3, 4, 4, and 5. Their use would result in the same overall project duration of 9.

7.3 Project Planning with Resources

So far, our discussion has centered around time planning. In the process, we have assumed that sufficient resources are available to perform the activities in the time specified for the individual activities. In this chapter we will extend the basic model by adding a resource requirement. For simplicity, we will use only a single resource, such as manpower, backhoes, machinery, or any other resource relevant to the project. For simplicity, we will refer to the resource as employees throughout this section. When we associate a resource requirement of, say, 30 units to an activity, then that means that we will need those 30 resource units throughout the duration of the activity. As an illustration, consider again the project network in Figure 7.3 in Section 7.1.

Furthermore, assume that the resource consumptions of the individual activities are shown in Table 7.5.

Table 7.5: Resource consumption of the activities in the example

Activity	*A*	*B*	*C*	*D*	*E*	*F*	*G*	*H*	*I*	*J*
Resource consumption	10	20	40	20	25	30	10	25	20	25

In order to schedule the activities of the project network, it is useful to employ a so-called *Gantt chart*. In essence, it is a horizontal bar chart, which features the individual activities on the ordinate, while the abscissa is a time axis. Clearly, the activities on the critical path are scheduled from their earliest possible (or, equally, latest allowable) start times, so that they form a non-overlapping sequence of bars that has no gaps. The black bars in Figure 7.10 belong to critical activities and their position in the graph cannot be changed.

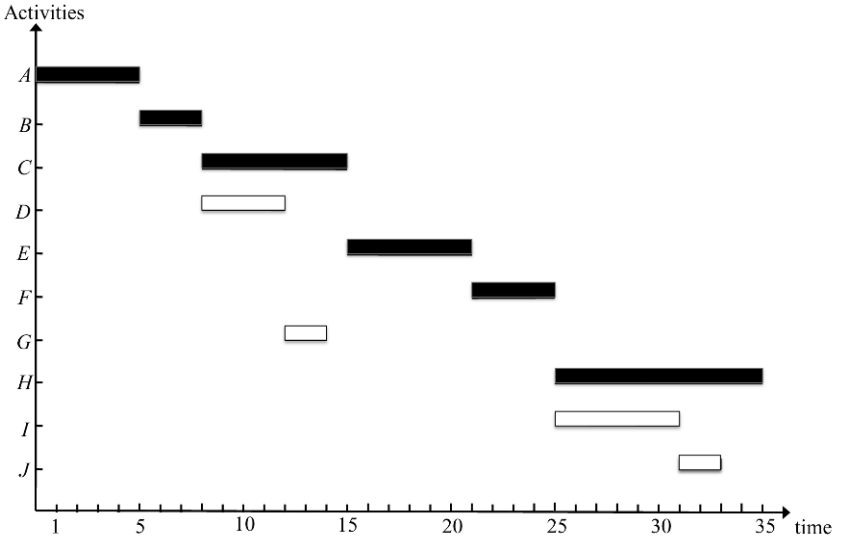


Figure 7.10

The matter is different with the noncritical activities, which have some leeway for their schedule. Suppose now that we use a heuristic method for scheduling them, which includes a rule that states that all noncritical activities should be scheduled a early as possible. The regular bars that belong to the activities *D*, *G*, *I*, and *J* in Figure 7.10 show how these activities are scheduled. This schedule now has very clear resource implications. From time $t = 0$ to $t = 5$, we only perform activity *A*, so that we need 10 employees. From $t = 5$ to $t = 8$, we perform only activity *B*, which requires 20 employees. Starting at $t = 8$ to time $t = 12$, we perform the activities *C* and *D* simultaneously. This requires 40 + 20 = 60 employees. At time $t = 12$, activity *D* is finished, while *C* is still going on. However, at $t = 12$, activity *G* is also scheduled, so that 40 employees for *C* and 10 employees for activity *G* are needed. This process continues until the project is finished. The resource requirements are shown in the *resource requirement graph* in Figure 7.11.

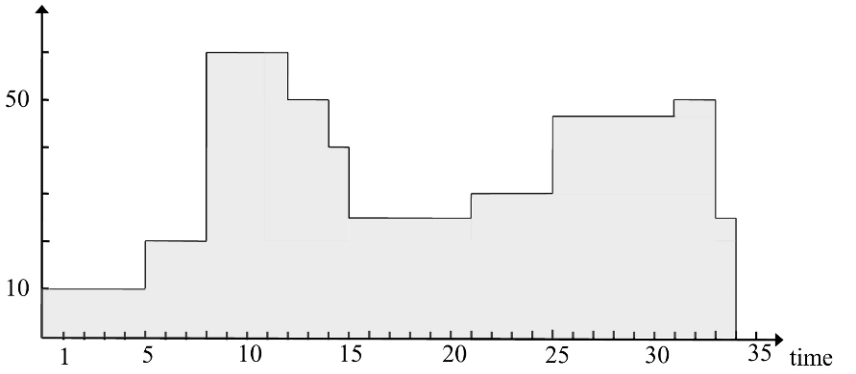


Figure 7.11

It is apparent that the resource requirement is very low in the beginning, then peaks, drops and increases again towards the end. If we were to be able to employ only casual labor that we need to pay only when needed, then the total resource requirement is the shaded area in Figure 7.11. Calculating the size of the area from the beginning we have 10 employees needed for 5 weeks, 20 employees needed for 3 weeks, 60 employees needed for 4 weeks, and so forth, for a total of 1,155 employee weeks. Assuming that we pay employees $15 per hour for 8 hours a day and 5 days a week, each employee will cost us $600 (plus fringe benefits, which we ignore here for simplicity). This means that the resource costs will be $693,000 for the entire project.

The situation changes dramatically if we have to hire all needed employees for the entire duration of the project. As the highest manpower requirement at any point in time is 60, this is the smallest number of permanent employees required for the duration of the project. Employing 60 employees for the total of 34 weeks at a cost of $600 per week costs $1,224,000, more than 76% more than the costs for casual labor. This is caused by the fact that more than 43% of the time the employees are paid, they are actually idle.

This calls for a different schedule whose maximal resource requirement is as low as possible. This type of objective is of the minimax type, where we search to minimize the maximum resource required at any point in time. Rather than using the heuristic that schedules all activities as early as possible (not a bad choice in general, as it allows for some noncritical activities to increase in duration without jeopardizing the finishing of the project on time), we will use another heuristic that schedules all activities as late as possible. The Gantt diagram that belongs to that schedule and the associated resource consumption graph are shown in Figure 7.12.

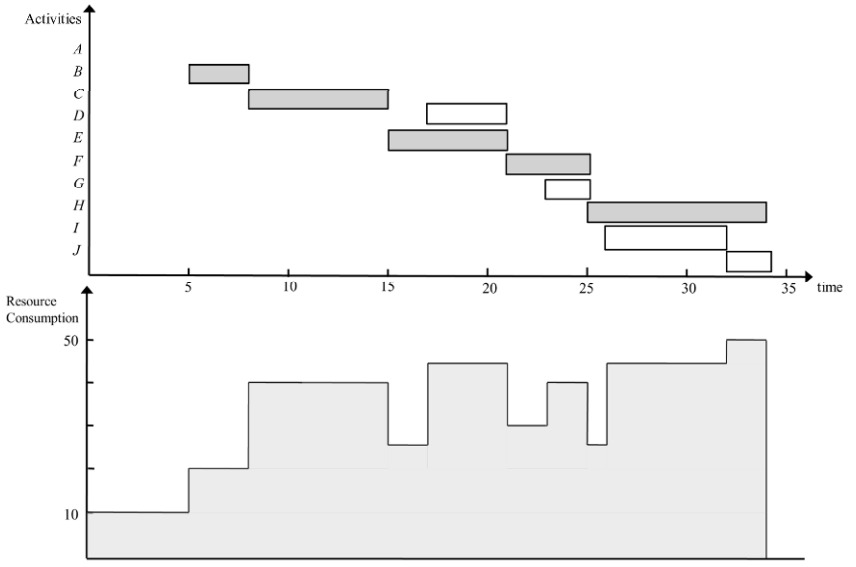


Figure 7.12

It turns out that while this schedule uses the exact same number of employee weeks of casual labor—1,155—the highest employee requirement at any one point is only 50. This means that the costs for employees working throughout the project is $1,020,000, which is 16.67% less than with the "earliest possible" schedule. The idle time here is still 32%, though.

Other heuristics exist for the scheduling of noncritical activities. Depending on the problem, they may be able to reduce the number of required resources further. The problem can also be solved by exact methods, but the integer programming problem that must be formulated for that purpose, is quite difficult. For details, see, e.g., Eiselt and Sandblom (2004).

7.4 The PERT Method

All project planning models discussed so far have in common that they are deterministic. More specifically, they have assumed that all components of the network—the activities, their durations, and the precedence structure—are known with certainty. This section will change that. In particular, we assume here that the activity durations are no longer known with certainty. It is important to realize that this is only one component that can be probabilistic: fully stochastic networks are dealt with by very sophisticated project network tools such as *GERT* (*graphical review and evaluation technique*), which are beyond the scope of this book.

The *PERT* method discussed in this section assumes that the duration of the activities are random variables with known underlying probability distributions. We do assume that the durations of activities are independent of each other. this is a fairly strong assumption, not justified in cases such as construction, where occurrences such as bad weather will affect many of the activities to take longer than they normally would. As usual in all of operations research, check the assumptions carefully, and if the assumptions do not fit the scenario under consideration, don't use the model.

Traditionally, it has been assumed that the duration of a single activity follows Euler's beta distribution. This assumption has been much criticized in the literature. However, we can derive the same formulas without making such a strict and controversial assumption. Our assumption is that the activity durations follow some bell-shaped distribution. For bell-shaped distributions, the *empirical rule* in statistics is known to apply; see Appendix D. It states that about all observations are within three standard deviations about the mean, while about two thirds of all observations are within one standard deviation about the mean. This leaves 1/6 of the total mass for each of the two tails of the distribution. This situation is shown in Figure 7.13.

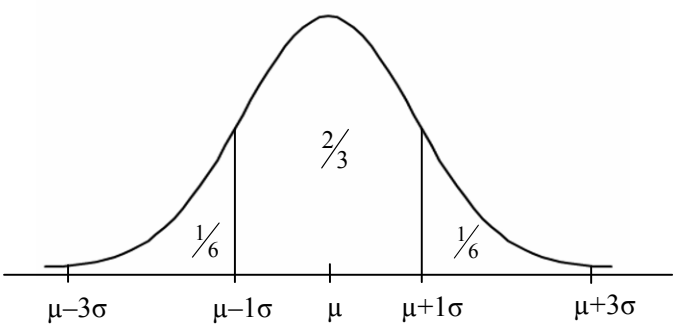


Figure 7.13

We can then define three time estimates for the duration of each activity: a most likely time estimate t_m (the mode of the distribution), a *pessimistic estimate* t_p, and an *optimistic estimate* t_o. The most likely time is associated with the central part of the distribution, the pessimistic estimate belongs to the right tail of the distribution, and the optimistic estimate is on the left tail of the distributions. Their weights are 2/3, 1/6, and 1/6 as shown in Figure 7.13. Based on these estimates, we can than compute a (weighted) mean for the duration of an activity as $t = \frac{1}{6}t_o + \frac{2}{3}t_m + \frac{1}{6}t_p = \frac{t_o + 4t_m + t_p}{6}$, which is exactly what was obtained by using the much stronger assumption of the beta distribution. Similarly, with $t_p = \mu + 3\sigma$ and $t_o = \mu - 3\sigma$, we can determine the variance of the activity duration as $\sigma^2 = \frac{1}{36}(t_p - t_o)^2$.

Armed with this information, we can then determine the mean duration and its variance for all of the given activities from the three time estimates, specified for each of them. Given the mean activity durations, we can use them in exactly the same way we dealt with time estimates in the critical path method. Going through the regular procedure—forward sweep, backward sweep, computation of floats, critical path—we can determine the critical path on the basis of the mean durations of the activities. In addition to the usual information about project duration, critical and noncritical activities, and sensitivity analyses on the basis of floats, we can use the variances of the durations on the critical path to make probability statements. In particular, we can provide the decision maker with an estimate concerning the probability with which the project can be completed within a certain time.

In order to explain the concept, consider a numerical example, whose numerical information is provided in Table 7.6. The graph for this project is the same as that for the examples in the previous two sections, but the time estimates are obviously different.

Table 7.6: Numerical information for example

Activity	Immediate predecessor	Time estimates (in hours)		
		optimistic	most likely	pessimistic
A	–	5	6	7
B	*A*	3	3	3
C	*A*, *B*	5	7	9
D	*B*	3	4	11
E	*B*, *C*	3	6	9
F	*C*, *D*, *E*	4	4	4
G	*D*	1	2	3
H	*F*, *G*	6	9	12
I	*F*, *G*	4	6	8
J	*I*	2	2	2

Our first task is to calculate the mean activity durations for the ten activities. They are 6, 3, 7, 5, 6, 4, 2, 9, 6, and 2, respectively. Those time estimates are then used in the standard procedure discussed in the first section of this chapter. The results are shown in Figure 7.14.

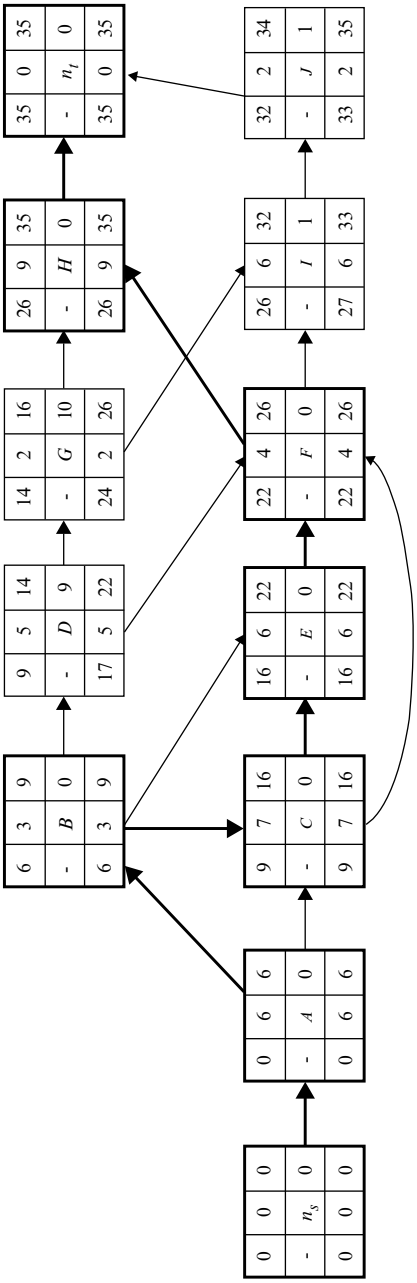

0 0 0
- n_S 0
0 0 0
0 6 6
- A 0
0 6 6
6 3 9
- B 0
6 3 9
9 5 14
- D 9
17 5 22
14 2 16
- G 10
24 2 26
26 9 35
- H 0
26 9 35
35 0 35
- n_I 0
35 0 35
9 7 16
- C 0
9 7 16
16 6 22
- E 0
16 6 22
22 4 26
- F 0
22 4 26
26 6 32
- I 1
27 6 33
32 2 34
- J 1
33 2 35

Figure 7.14

Consider now the critical path $n_s - A - B - C - E - F - H - n_t$. The mean project duration of $\mu = 35$ has already been computed in the procedure. We now have to calculate the variance on this path. The variances of the individual activities on the path (in order of their appearance) are $\frac{4}{36}, 0, \frac{16}{36}, \frac{36}{36}, 0,$ and $\frac{16}{36}$. The sum of these variances equals $\sigma^2 = \frac{92}{36}$, so that the standard deviation equals $\sigma = \sqrt{\frac{92}{36}} \cong 1.5986$. Furthermore, the central limit theorem states that the project duration is approximately normal with mean μ and standard deviation σ.

Given this information, we are now able to provide the decision maker with some rough estimate about the probability with which the project can be finished within a prespecified time frame T. Suppose that the decision maker wants to know what the probability is that the project is finished within $T = 36$ hours. In order to be able to use the standard normal distribution (see Appendix D in this book) we calculate the z-score as $z = \frac{T - \mu}{\sigma} = \frac{36 - 35}{1.5986} \cong 0.6255$, we find that the probability $P(X \leq 36) = 73.42\%$. When calculating these probabilities, it is always useful to draw the normal distribution function and indicate which area we are looking for. The area relevant to this question is shown in Figure 7.15a, where it constitutes the shaded area plus the entire mass to the left of the mean which, by definition, equals 0.5.

Similarly, we could compute the probability that the project takes more than 37 hours. Such information may be needed by the decision maker, as the late completion of the project may carry a penalty with it. The z-score for the completion time of 37 is $z = \frac{37 - 35}{1.5986} \cong 1.2511$, from which we obtain a probability of $P(X \geq 37) = 10.58\%$. The area of interest under the normal distribution function is shown in Figure 7.15b.

Finally, we compute the probability that the project is completed between 33 and 37 hours. The area of interest is shown in Figure 7.15c, and it can be computed as $P(33 \leq X \leq 37) = P(33 \leq X \leq 35) + P(35 \leq X \leq 37) = 0.7884$. In other words, chances are about 80% that our project will be completed within the specified time window.

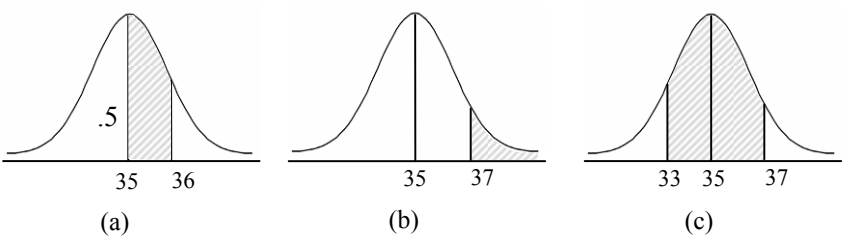


Figure 7.15

A few comments are in order. First of all, much care should be taken with the probability statements. They have been derived with a lot of assumptions, so that it makes absolutely no sense to report them to the decision maker with two, four, or even more digits to the right of the decimal point, implying great accuracy. These probabilities are rough estimates, and this should be emphasized throughout.

Secondly, what we have used is what is called the time-critical path. In other words, we have determined the critical path on the basis of estimated activity durations only, and then computed the probabilities. This does not necessarily provide the planner with the true critical path. As an example, consider two paths in some network, one with a mean duration $\mu = 100$ and a standard deviation of $\sigma = 10$, while a second, obviously noncritical, path exists in the network with $\mu = 99$ and $\sigma = 100$. Note that the noncritical path is shorter, but has a much higher standard deviation. The probability to finish the project, computed on the basis of the critical path as we do, is then 84.13%. However, computing the same probability on the basis of the noncritical path is only 54.38%, meaning that the former result (that we would obtain with our procedure), grossly overestimates the likelihood to finish the project within the specified time frame. This problem persists in a somewhat different guise even in our example: all probability statements assume that the critical path remains critical. Note, however, that the noncritical activities *I* and *J* have a very small float, making them almost critical. If their durations were to increase by fairly small, insignificant amounts, they would become critical, and their standard deviations, which have been completely ignored so far, would suddenly have to be counted. This is yet another reason to treat the probability statements we have calculated with the utmost caution.

Exercises

Problem 1 (acceleration of a project): Consider a project network with four activities, their normal durations, their shortest possible durations (which can be achieved at extra cost), & the acceleration cost per time unit. Details are shown in Table 7.7.

Table 7.7: Details of the network in Problem 1

Activity	Immediate predecessor(s)	Normal duration (in days)	Shortest possible duration (in days)	Unit cost of acceleration
A	–	4	3	80
B	*A*	5	3	30
C	*A*	3	2	50
D	*A*, *C*	2	1	40

(a) Draw the project network.
(b) Use the forward recursion, the backward recursion, calculate the floats, and determine the critical path(s). What is the duration of the project?
(c) Accelerate the project by one day. Clearly indicate which (combinations of) activities could be accelerated in order to speed up the project, and which activity or activities should be accelerated. What are the associated costs?
(d) What if the decision maker needs some further project acceleration? What is the shortest project duration and what are the associated costs?

Solution:

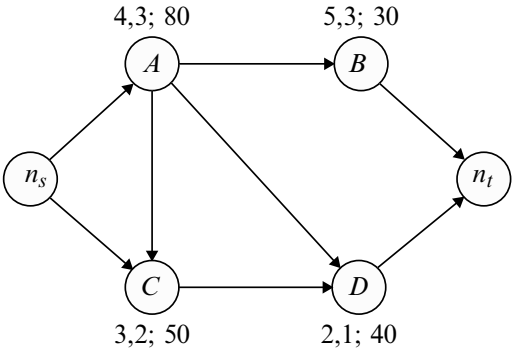


Figure 7.16

(d) After the first acceleration (c), the activity durations are 4, 4, 3, and 1, and the project duration is $T = 8$. At this time, we either accelerate A (cost \$80) or activities B and C (cost \$80). Arbitrarily choose activity A. the activity durations are then 3, 4, 3, and 1, and the project length is $T = 7$. The two critical paths are still the only critical paths in the network. At this point, we can only accelerate activities B and C, as A and D are both at their respective crash times. Accelerating B and C costs \$80. This leads to a project duration of $T = 6$, which cannot be accelerated any further. The cost to get to this point is \$230.

Problem 2 (scheduling with resources, Gantt chart and resource consumption graph): A project has been subdivided into five activities. Their immediate predecessors, activity durations, and resource consumption are shown in Table 7.8.

Table 7.8: Details of the network in Problem 2

Activity	Immediate Predecessor	Duration (in days)	Resource consumption
A	–	3	40
B	–	7	40
C	*A*	5	30
D	*A*, *B*, *C*	2	60
E	*B*, *D*	5	70

(a) Draw the project network.
(b) Calculate all earliest and latest starting and finishing times and the total floats of all activities. What is the critical path? What is its duration?
(c) Draw the Gantt chart and the associated resource consumption graph, assuming that all activities are scheduled as early as possible. What is the largest resource consumption at any point in time?

Solution: (a)

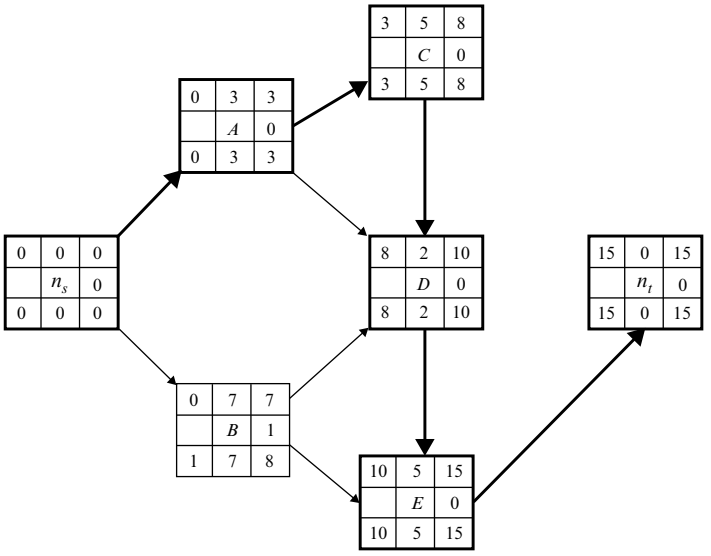


Figure 7.17

(b) The critical path includes the activities *A*, *C*, *D*, and *E*. Its length is 15.

(c)

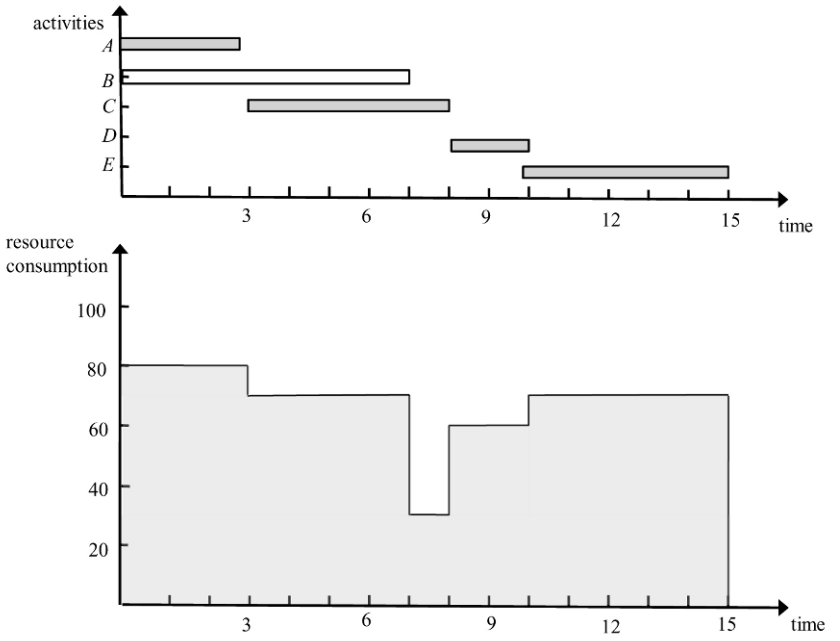


Figure 7.18

The largest resource consumption at any point in time is 80.

Problem 3 (PERT network): A project has been subdivided into five activities. Their immediate predecessors and the estimated activity durations (optimistic, most likely, and pessimistic) are shown in Table 7.9.

Table 7.9: Details of the network in Problem 3

Activity	Immediate Predecessor	Estimated duration (in days)
A	–	1, 2, 3
B	–	1, 3, 5
C	*A*, *B*	3, 4, 5
D	*A*, *C*	2, 2, 2
E	*C*, *D*	1, 2, 21

(a) Draw the project network. Based on the mean activity durations, calculate all earliest and latest starting and finishing times, and the total floats of all activities. What is the critical path? What is its duration?
(b) Calculate the variance and standard deviation on the critical path.
(c) Calculate the probability that the project will be finished between 11 and 15 days.

(d) What is the probability that the project will be finished within 14 days? What is the probability that the project will be finished in exactly 14 days?

Solution: (a)

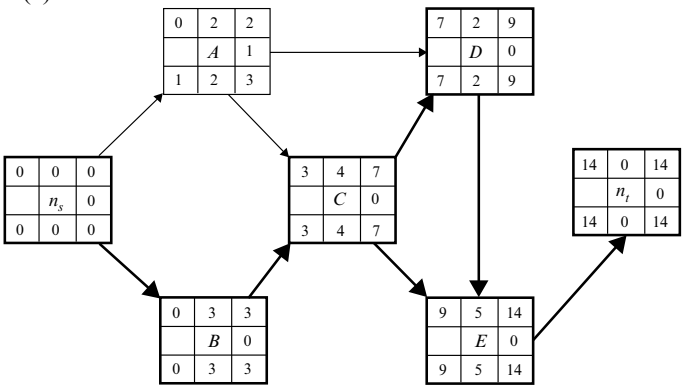


Figure 7.19

The critical path is $n_s - B - C - D - E - n_t$. Its duration is 14 days.

(b) $\sigma^2 = \frac{1}{36}[16 + 4 + 0 + 400] = \frac{420}{36}$, $\sigma = 3.4157$.

(c) $P(11 \le X \le 15) = P(11 \le X \le 14) + P(14 \le X \le 15)$. Finding z-scores z_1 and z_2 for the two ranges results in $z_1 = \dfrac{14-11}{3.4157} = .8783$, leading to 31.06%, and $z_2 = \dfrac{15-14}{3.4157} = .2928$, leading to 11.41%, for a total of 42.47%.

(d) $P(X \le 14) = 0.5$. $P(X = 14) = 0$.

Problem 4 (GANTT chart, resource requirement graph, and PERT network): A project has been subdivided into five activities. Their immediate predecessors and the estimated activity durations (optimistic, most likely, and pessimistic) are shown in Table 7.10.

Table 7.10: Details of the network in Problem 4

Activity	Immediate Predecessor	Estimated duration (in days)
A	–	2, 6, 16
B	–	2, 2, 8
C	*A*	6, 8, 10
D	*B*	5, 6, 7
E	*A*, *B*	9, 9, 9

(a) Draw the project network. On the basis of the expected durations, calculate all earliest and latest starting and finishing times, and the total floats of all activities. What is the critical path? What is its duration?
(b) Calculate the variance and standard deviation on the critical path.
(c) Calculate the probability that the project will be finished between 14 and 18 days.
(d) What would happen to the result under (c), if the time estimates of activity *D* were to be revised to 3, 6, and 9? Explain in one short sentence.
(e) Consider the starting and finishing times calculated in (a) as well as resource requirements of 20, 50, 30, 40, and 60, respectively. On that basis, draw a GANTT diagram given that all activities are scheduled as early as possible. What is the highest resource requirement at any one time during the project?
(f) Repeat question (e) for the "Latest possible" scheduling rule.

Solution: (a)

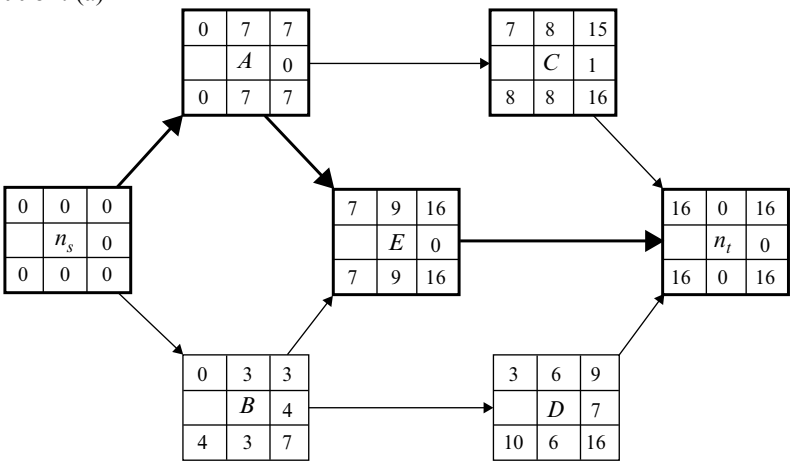


Figure 7.20

The critical path is $n_s - A - E - n_t$, and its length is 16.

(b) and (c) $\sigma^2 = \frac{1}{36}(196 + 0) \approx 5.4444$ and $\sigma \approx 2{,}3333$. Then $z = \dfrac{18-16}{2.3333} = .8572$, and $P(14 \le X \le 18) = 2\,(.3043) = 60.86\%$.
(d) Since activity *D* is not on the critical path, there will be no changes.

(e)

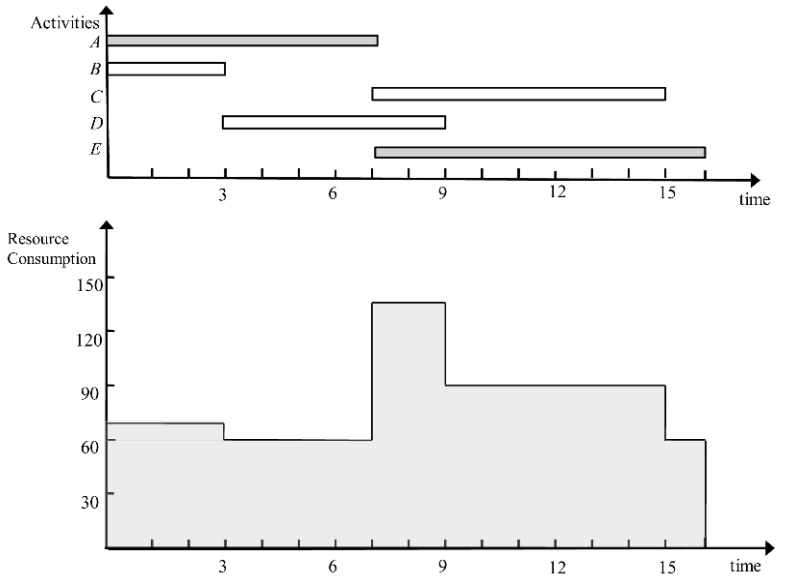


Figure 7.21

The highest resource requirement at any point in time during the project is 130.

(f)

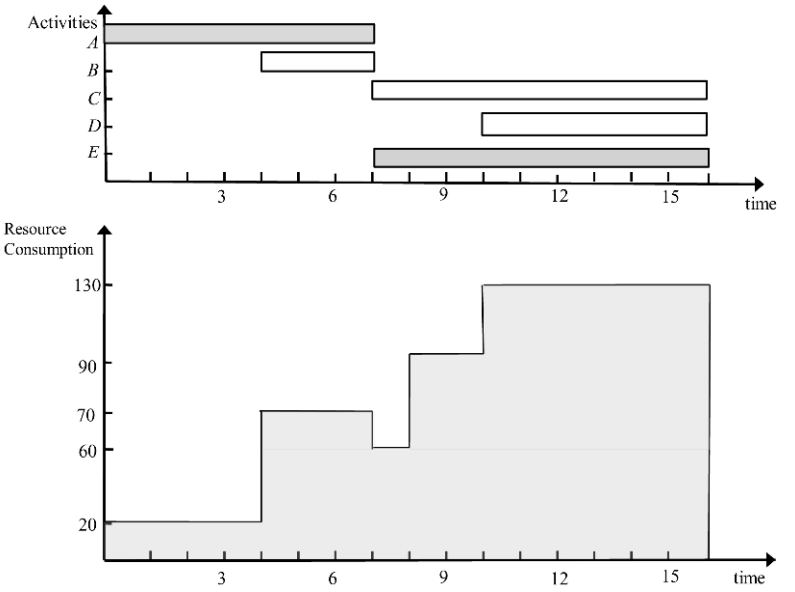


Figure 7.22

8 Machine Scheduling

The subject of this chapter is the allocation of *jobs* (or *tasks*) to *processors* (or *machines*). These terms should be understood in the widest possible sense: in the case of a doctor treating patients the doctor is the processor and the patients represent the tasks, in the case of a tax auditor the auditor is the processor (or the "machine"), while the individual cases are the tasks to be processed. This allocation is to be made so as to optimize some objective. We will discuss a number of these objectives below.

This problem is somewhat reminiscent of the production scheduling problem introduced in Chapter 2 of this volume, and the project planning problems discussed in Chapter 7. The main feature of the machine scheduling problems in this chapter is that they include a *sequencing* component, which determines the *order* in which tasks are processed on a machine, and a *scheduling* component that determines at what *time* the processing of a task begins and ends on a machine. Often, the term scheduling is meant to include both the sequencing and scheduling parts of the allocation problem.

This chapter will deal exclusively with deterministic scheduling problems, i.e., situations in which all parameters are assumed to be known with certainty. This is not to mean that these are the only relevant scheduling problems: on the contrary, in many important real-world applications, some of the parameters involved in the problem are uncertain. However, the structure of probabilistic problems is no different from that of deterministic problems (other than the fact that probabilistic problems are typically much more difficult than their deterministic counterparts), and what we attempt to convey in this chapter is a general idea of scheduling models, their applications, and degree of difficulty.

The first section of this chapter will introduce the basic concepts of scheduling models. The remaining three sections of this chapter deal with different scheduling scenarios with increasing degree of difficulty.

H.A. Eiselt and C.-L. Sandblom, *Operations Research: A Model-Based Approach*,
DOI 10.1007/978-3-642-10326-1_8, © Springer-Verlag Berlin Heidelberg 2010

8.1 Basic Concepts of Machine Scheduling

As mentioned above, each machine scheduling problem features n tasks (jobs) T_1, T_2, …, T_n that are to be performed on m machines (processors) P_1, P_2, …, P_m. Processing a task on a machine will take a certain (and known) amount of time, which is referred to as the *processing time*. In particular, we define p_{ij} as the processing time of task T_j on machine P_i for all combinations of tasks and machines. In case there is only a single machine or all machines have the same processing time for a job, we simplify the notation to p_j.

We can then distinguish between three broad categories of models. The first category assumes that there is only a single machine, we refer to this as *single machine scheduling*. The second category has multiple machines working in parallel, all able to perform the same function. This type of model is called a *parallel machine scheduling problem*. There are two subcategories of parallel machine scheduling. In the first, parallel machines are *identical* (meaning that all machines take the same amount of time to process any given task), or they are *unrelated*, which indicates that it takes different amounts of time to process any given task on different machines.

The third broad category of models includes *dedicated machine scheduling models*. The main characteristic of these models is that not all machines have the capability to process all jobs, and not all jobs need to be processed on all machines. In particular, we distinguish between three subcategories. The first category is an *open shop*. In an open shop, each task must be processed by all machines, but there is no specific order in which the processing must take place. The second category are *flow shops*, in which each job is to be processed on all machines, and each task is processed by the machines in the same specified order. Finally, there are *job shops*, in which each job needs to be processed on a specific set of machines, and the processing order is also job-specific.

In addition to the processing time p_j introduced above, we may also have a number of additional parameters. The first such parameter is the *ready time* (also referred to as *arrival time* or *release time*) r_j, which indicates the time at which task T_j is ready for processing. In the simplest case, $r_j = 0$ for all tasks T_j, meaning that all jobs are ready when the scheduling process starts. The second additional parameter is the *due time* d_j, which is the time at which task T_j should be finished. This could be a time specified in a contract or some other delivery time that was agreed upon. If the task is not completed by that time, then there may be a penalty for the delay.

Once all jobs have been scheduled on the machines, we can use a number of different properties inherent in any given schedule. The first is the *completion time* c_j of job T_j, which is the time at which task T_j is completely finished. The second property is the *flow time* f_j of task T_j. The flow time is formally defined as $f_j = c_j - r_j$, and it can be thought of as the time that a job is in the system, either waiting to

be processed or being processed. (This time is reminiscent as the "time in the system" W_s in the analysis of waiting lines, see Chapter 12 of this book). Finally, there is the *lateness of a task* T_j, which is defined as $\ell_j = c_j - d_j$ and *tardiness*, which is expressed as $t_j = \max\{\ell_j, 0\}$. The lateness of a task is the time that elapses after the due date and the actual completion date of a task. For late jobs, lateness and tardiness are the same. For jobs that are completed before their due date, the lateness becomes negative, while their tardiness equals zero.

There are many different criteria that may be used to optimize scheduling systems. Three of the most popular such criteria are introduced here. First, there is the *makespan* (or *schedule length*) C_{max}. It is formally defined as $C_{max} = \max_j \{c_j\}$, and it expresses the time at which the last of the tasks has been completed. Such a measure is meaningful if a project can only be considered completed, if all of its individual tasks have been completed (similar to project networks, see Chapter 7 of this volume). A typical example is the processing of machine parts that have been ordered by a customer. All of them will be shipped in a box, and the box can only be released for transportation, once all of the individual machine parts are included.

The second criterion is the *mean flow time F*. The mean flow time is formally defined as the unweighted average $F = \frac{1}{n}(f_1 + f_2 + ... + f_n)$. (As a matter of fact, by virtue of the definition of flow time, it is easy to demonstrate that the mean flow time differs from the mean completion time $\frac{1}{n}(c_1 + c_2 + ... + c_n)$ only by the constant $\frac{1}{n}(r_1 + r_2 + ... + r_n)$). The mean flow time refers to the average time that a job is in the system either waiting to be processed or being processed. The mean flow time is a meaningful measure in instances such as a maintenance or repair system, in which a machine is not available if it is waiting for repair or being repaired.

The third and last criterion in this context is the *maximal lateness* L_{max}. The maximal lateness is defined as $L_{max} = \max_j \{\ell_j\}$, and it expresses the longest lateness among any of the jobs. This criterion is applicable in case some lateness is unavoidable or deemed acceptable, but the decision maker attempts to ensure that very long delays beyond the due date are avoided.

Before discussing any details of specific scheduling models, we should point out some general features inherent in scheduling problems. In this type of model there is a fairly fine line that separates models that are rather easy to solve (some of which are represented in this chapter), while others, seemingly straightforward extensions, are very difficult. In those cases, exact methods that find optimal solutions will take a very long time, which may—depending on the individual circumstances—render them impractical. In such cases, decision makers will

resort to heuristic methods that have the advantage of providing (hopefully good) solutions quickly, but the obvious disadvantage of not necessarily resulting in an optimal solution. If a difficult problem has to be solved in real time, the use of a heuristic is imperative; if some time is available, an exact algorithm may be employed.

8.2 Single Machine Scheduling

The models in this section deal with the simplest of scheduling problems: there is only a single machine on which tasks are to be processed. Before investigating the solutions that result from the use of the three criteria introduced in the introduction above, we may introduce another wrinkle in this seemingly primitive scenario. In particular, we may consider modes that allow the *preemption* of a task, while others do not. In case preemption is permitted, this means that each task has associated with it a priority, and if a task with a higher priority becomes available at a time, when a task with a lower priority is being processed, then processing on the lower-level task stops, and the higher-level task is processed first. Examples of preemption abound: consider the case of a surgeon who is dealing with a broken leg as another patient with a heart attack arrives. Rather than referring to the usual "first come, first served" rule (inviting juicy lawsuits), most surgeons would probably stabilize the broken leg and deal with the heart patient first. Similar preemptions are found for police officers, who would interrupt a routine investigation to attend to a robbery in progress, or a plumber, who will interrupt the installation of a water pump in a residence to attend to a broken main. In this section, we restrict ourselves to cases, in which preemption is not permitted.

Minimizing the makespan in case of a single machine is not meaningful, as each sequence of tasks will result in the very same value of C_{max}. More specifically, C_{max} equals the sum of processing times of all tasks.

The objective that minimizes the mean flow time is not as straightforward. However, it is not difficult either, as it has been shown that the simple *Shortest Processing Time* (*STP*) *Algorithm* solves the problem optimally. The algorithm can be summarized by a simple rule.

> *SPT Algorithm*: Schedule the task with the shortest processing time first. Delete the task and repeat the procedure until all tasks have been scheduled.

Rather than illustrate the *SPT* algorithm by an example, we will first introduce a minor extension of the rule. In particular, suppose that the decision maker has not only processing times p_j to consider, but there are also weights w_j associated with the tasks. The objective is then to minimize the average weighted flow time, defined for task T_j as $w_j f_j$. The weighted generalization of the *SPT* algorithm can than be stated as

> *WSPT Algorithm* (*Smith's Ratio Rule*): Schedule the task with the shortest weighted processing time p_j/w_j first. Delete the task and repeat the procedure until all tasks have been scheduled.

As a numerical illustration, consider

Example 1: There are seven machines in a manufacturing unit. Scheduled maintenance has to be performed on these machines once in a while. The repairman has identified the processing times required for the maintenance. Costs are incurred for downtime, regardless if the machine waits for service or is being served. These costs differ between the machines. The estimated processing times of the machines, the downtime costs, and the weighted processing times are summarized in Table 8.1.

Table 8.1: Processing times and downtime cost for Example 1

Job #	T_1	T_2	T_3	T_4	T_5	T_6	T_7
Service time (minutes)	30	25	40	50	45	60	35
Downtime cost ($ per minute)	2	3	6	9	4	8	3
Weighted processing time p_j/w_j	15	8⅓	6⅔	$5\frac{5}{9}$	11¼	7½	11⅔

Applying the *WSPT* algorithm, we first schedule task T_4 (which has the lowest weighted processing time of $5\frac{5}{9}$), followed by T_3 with the next-lowest weighted processing time of 6⅔, followed by T_6, T_2, T_5, T_7, and T_1. The schedule is shown in the *Gantt chart* (named after the American engineer Henry L. Gantt (1861 – 1919), who developed these charts in 1917) in Figure 8.1.

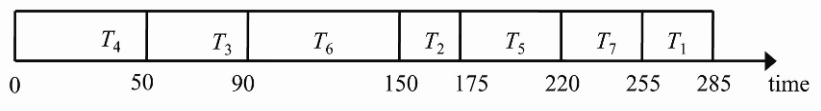


Figure 8.1

The tasks T_4, T_3, T_6, …, T_1 now have idle times of 0, 50, 90, 150, 175, 220, and 255, respectively. Adding the processing time to them results in 50, 90, 150, 175, 220, 255, and 285, respectively. Multiplying these by the individual per-minute costs and adding them up results in a total of $4,930.

Consider now the objective that minimizes maximal lateness L_{max}. Again, this problem turns out to be easy from a computational point of view. A simple

method was developed in the mid-1950s by Jackson, which is now commonly referred to as the *earliest due date algorithm* (or *EDD algorithm* or *Jackson's rule*). It finds an optimal solution and can be stated as follows.

> *EDD Algorithm*: Schedule the task with the earliest due date first. Delete the task and repeat the procedure until all tasks have been scheduled.

We will explain this rule by means of

Example 2: The accounting department of a large firm processes book-keeping jobs for various divisions of the firm. At present, they have identified eleven tasks, which are to be completed by a single team, one after another. The processing times and the due dates for the individual jobs are shown in Table 8.2.

Table 8.2: Data for Example 2

Job #	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	T_9	T_{10}	T_{11}
Processing time (hours)	6	9	4	11	7	5	5	3	14	8	4
Due dates	25	15	32	70	55	10	45	30	30	80	58

The *EDD* rule starts by scheduling task T_6 first (its due date is 10, the earliest of all due dates), followed by T_2, T_1, and so forth. The Gantt chart for the schedule obtained by the *EDD* rule is shown in Figure 8.3 (where the tie between tasks T_8 and T_9 that have the same due dates, is broken arbitrarily).

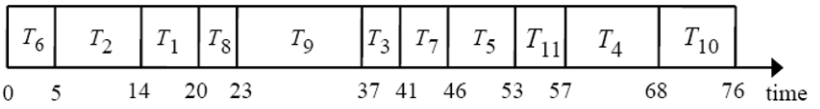


Figure 8.2

Simple inspection reveals that the tasks T_6, T_2, T_1, and T_8 are finished before the due date. Task T_9 is late by 7 hours, T_3 is late by 9 hours, T_7 is late by 1 hour, T_5, T_{11}, T_4, and T_{10} are again finished before their due dates. This means that the maximal lateness occurs for job T_3, so that $L_{max} = 9$. Had we broken the tie in favor of T_9 rather than T_8, task T_9 would have been completed at time 34 (4 hours late), task T_8 would have been finished at time 37 (7 hours late). Otherwise, the schedule would have been identical to that shown in Figure 8.2, with $L_{max} = 9$ being still defined by job T_3.

The discussion in this chapter may leave readers with the impression that single-machine scheduling problems are easy. This is, however, not the case. Consider again Example 2, but as an objective, we now use the total tardiness rather than

the maximal lateness. In other words, our objective is now $L_{sum} = t_1 + t_2 + \ldots + t_{11}$ rather than $L_{max} = \max\{\ell_1, \ell_2, ..., \ell_{11}\}$. We can find an optimal solution to the problem by formulating an integer programming problem. In order to do so, we first use variables t_1, t_2, …, t_{11} for the tardiness of the individual tasks. We then define variables x_1, x_2, …, x_{11} for the actual starting times of the eleven tasks. Furthermore, we need to define zero-one variables y_{ij} as

$$y_{ij} = \begin{cases} 1, \text{ if job } T_i \text{ directly precedes job } T_j \\ 0, \text{ otherwise} \end{cases},$$

where the subscripts i and j each can assume any value between 1 and 11, resulting in $11^2 = 121$ zero-one variables y_{ij} in addition to the eleven continuous variables t_1, …, t_{11} and the eleven continuous variables x_1, …, x_{11} for a total of 143 variables.

The objective simply minimizes the sum of tardinesses $t_1 + t_2 + \ldots + t_{11}$. As far as constraints are concerned, we first must define tardiness, which is done by stating that tardiness of a job can be expressed as its starting time plus its processing time (resulting in its finishing time) minus the due date, assuming that the resulting figure is positive. As an example, consider job T_1, whose lateness can be written as $t_1 = x_1 + p_1 - d_1$, provided that $t_1 \geq 0$, otherwise t_1 is set to zero. This can be achieved by writing $x_1 + p_1 - d_1 \leq t_1$, as whatever value the left-hand side of this inequality assumes, the objective function will ensure that the value of t_1 is chosen as small as possible. This constraint is written for each of the eleven tasks separately.

We then have to ensure that a job must be completely finished before the next job in line can begin being processed. Suppose that we have scheduled job T_1 directly before job T_2. The pertinent constraint will then be $x_1 + p_1 \leq x_2 + M(1-y_{12})$, where $M >> 0$ is a suitably chosen large number. This constraint can be explained as follows. The left-hand side expresses the finishing time of job T_1, while the right-hand side shows the starting time of job T_2 plus M, if T_1 is not scheduled before T_2 and 0 if it is. In other words, if T_1 is scheduled before T_2, then this constraint requires that the starting time of T_2 is at least as large as the finishing time of T_1 (i.e., T_2 cannot start before T_1 is finished), while in case that T_1 is not scheduled before T_2, then the right-hand side of the inequality is very large, ensuring that it is satisfied regardless of the values of the variables. The problem can then be written as follows:

$$\begin{aligned} &\text{P: Min } z = t_1 + t_2 + \ldots + t_{11} \\ &\text{s.t. } x_1 + p_1 - d_1 \leq t_1 \\ &\quad \vdots \\ &x_{11} + p_{11} - d_{11} \leq t_{11} \\ &x_1 + p_1 - x_1 \leq M(1-y_{11}) \end{aligned}$$

$$x_1 + p_1 - x_2 \le M(1-y_{12})$$
$$\vdots$$
$$x_1 + p_1 - x_{11} \le M(1-y_{1,11})$$
$$x_2 + p_2 - x_1 \le M(1-y_{21})$$
$$\vdots$$
$$x_{11} + p_{11} - x_{11} \le M(1-y_{11,11})$$
$$x_1, x_2, \ldots, x_1 \ge 0$$
$$t_1, t_2, \ldots, t_{11} \ge 0$$
$$y_{11}, y_{12}, \ldots, y_{1,11}, y_{21}, y_{22}, \ldots y_{11,11} = 0 \text{ or } 1.$$

Note that the size of the integer programming problem for a scheduling model with n tasks includes n^2 zero-one variables, $2n$ continuous variables, and $n+n^2$ structural constraints plus nonnegativity and zero-one conditions. It is apparent that the formulation becomes unwieldy for large numbers of tasks, thus possibly requiring the use of heuristic algorithms.

8.3 Parallel Machine Scheduling

All scheduling models in this section have in common that the tasks can now be processed on more than one machine. In general, we assume that a given number m of machines are available. We further assume that these machines are identical in the sense that not only can all machines process each of the tasks, but it takes the same amount of time to process a task, regardless of the machine it is processed on.

First consider the objective of minimizing makespan C_{max}. It can be demonstrated that this problem is very difficult from a computational point of view, even for just two machines. This means that we typically have to resort to heuristics to solve the problem (except in cases, in which there is ample time to find exact solutions). The most popular heuristic method for this type of problem is the *longest processing time first* (or *LPT*) algorithm. This heuristic method belongs to the class of *list scheduling methods*. All list scheduling methods first produce a priority list of tasks, starting with the job that is assigned the highest priority. Using this list and starting with the task that has the highest priority, jobs are then assigned one at a time to the first available machine. In particular, the longest processing time first algorithm can be described by the following rule:

> *LPT Algorithm*: Put the tasks in order of nonincreasing processing times. Starting at the top, assign the first available task to the first available machine. Repeat the procedure until all tasks have been scheduled.

In order to demonstrate the way the algorithm works, consider again Example 1 above, but assume that now we have three servicemen available to perform the

maintenance tasks. Putting the seven tasks in order of their processing time, starting with the longest, we obtain the sequence T_6, T_4, T_5, T_3, T_7, T_1, and T_2 with processing times of 60, 50, 45, 40, 35, 30, and 25 minutes. In the beginning, all three machines are available, so we first assign the longest task T_6 to machine P_1. (Note that this machine will become available again at time 60, when T_6 is completely processed). The next task in line is T_4, which is assigned to machine P_2, which is available now. (This machine will become available again at time 50, when T_4 is completely processed). The next task in line is T_5 and it is assigned to machine P_3. This machine will become available again at time 45. The next task to be scheduled is now T_3. The three machines become available again at 60, 50, and 45, so that assigning T_3 to the next available machine means that it is scheduled on P_3. This process continues until all jobs are scheduled. The actual schedule is shown in the Gantt chart in Figure 8.3, where the shaded areas indicate the idle time on the machines.

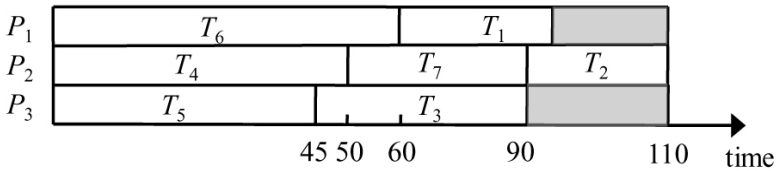


Figure 8.3

Note that the schedule length is C_{max} = 110. It is worth mentioning that this schedule is not optimal, which is not surprising, since the *LPT* algorithm, which it was determined with, is not an exact algorithm but a heuristic. Incidentally, the optimal solution schedules T_6 and T_7 on machine P_1, jobs T_4 and T_5 are processed on machine P_2, and tasks T_3, T_1, and T_2 are processed on P_3. This schedule has no idle time at all and all machines finish at time 95. Note that the optimality of a schedule does *not* necessarily require that there is no idle time. On the other hand, if there is no idle time, the schedule must obviously optimal.

Consider now the second of our objectives, which is to minimize mean flow time. It has been demonstrated that this problem can be solved to optimality by means of a fairly simple technique. The algorithm is frequently referred to as *McNaughton's rule*. It assumes that all tasks are ready at the beginning of the process, meaning that the mean flow time reduces to the mean completion time.

Recall that there are n jobs and m machines. We can then formulate the following method:

> *McNaughton's Algorithm*: First sort the jobs in order of nondecreasing processing time (ties are broken arbitrarily). Renumber them as $T'_1, T'_2, ..., T'_n$. Then assign tasks T'_1, T'_{1+m}, $T'_{1+2m}, ...$ to machine P_1, tasks $T'_2, T'_{2+m}, T'_{2+2m}, ...$ to machine P_2,

tasks $T'_3, T'_{3+m}, T'_{3+2m}, \ldots$ to machine P_3, and so forth. The tasks are processed in the order that they are assigned in.

In order to illustrate this algorithm, consider again Example 1 above. Recall that the reordered sequence of tasks is (T'_1, T'_2, T'_3, T'_4, T'_5, T'_6, T'_7) = (T_2, T_1, T_7, T_3, T_5, T_4, T_6) with the processing times (25, 30, 35, 40, 45, 50, 60). Given that we have three machines, we assign to machine P_1 the tasks T'_1, T'_4, and T'_7 (or, renumbering them again, T_2, T_3, and T_6), machine P_2 is assigned the jobs T'_2 and T'_5 (i.e., T_1 and T_5), and machine P_3 will process jobs T'_3 and T'_6, i.e., T_7 and T_4. The resulting optimal schedule is shown in Figure 8.4.

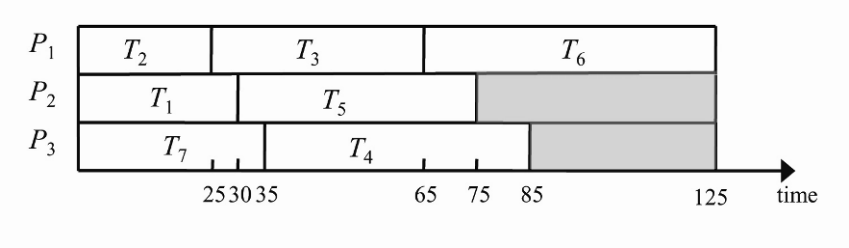


Figure 8.4

The mean completion time of the solution in Figure 8.5 is $F = \frac{1}{7}[25 + 65 + 125 + 30 + 75 + 35 + 85] = 440/7 \cong 62.8571$. The seemingly straightforward extension to the model with nonidentical ready times renders the problem very difficult from a computation point of view. This is yet another indication how fine a line there is between (sometimes very) easy and (sometimes very) difficult models.

Finally, the minimization of the maximal lateness in case of parallel machines turns out to be difficult, and we leave its discussion to specialized books, such as Eiselt and Sandblom (2004).

8.4 Dedicated Machine Scheduling

This section deals with different types of dedicated machine scheduling models. The first such model includes an open shop. Recall that in an open shop, each task must be processed on each of a number of different machines, performing different operations. The sequence of machines, in which the jobs are processed, is immaterial. Here, we will deal only with the case of two machines, which happens to be easy, while problems with three or more machines are difficult. Minimizing the schedule length (makespan) C_{max} is easy. Optimal schedules can be found by means of the *Longest Alternate Processing Time* (*LAPT*) *Algorithm*. It can be described as follows.

> *LAPT Algorithm*: Whenever a machine becomes idle, schedule the task on it that has the longest processing time *on the other machine*, provided the task has not yet been processed on that machine and is available at that time. If a task is not available, the task with the next-longest processing time on the other machine is scheduled. Ties are broken arbitrarily.

In order to illustrate the method, consider

Example 3: In an automotive assembly shop, each semi-finished product goes through two phases, assembly of individual components, and checking of the components. The sequence in which these tasks are performed is immaterial. The times (in minutes) that it takes to assemble and check the six products are shown in Table 8.3.

Table 8.3: Data for Example 3

Job #	T_1	T_2	T_3	T_4	T_5	T_6
Processing time on P_1	30	15	40	30	10	25
Processing time on P_2	35	20	40	20	5	30

Using the *LAPT* algorithm, we begin by scheduling a task on machine P_1. The task with the longest processing time on P_2 is T_3, so this job is scheduled first on P_1. The next step is to schedule a task on P_2 which now (we are still at time zero) idle. The task with the longest processing time on P_1 is again T_3, which is not available at this time, so that we schedule the task with the next-longest processing time on P_1 next. This is either T_1 or T_4. Arbitrarily choose T_1. With tasks T_3 and T_1 being scheduled on the two machines, T_1 is the first task whose processing is finished at time 35, and machine P_2 becomes idle again. At this time, the available task with the next longest processing time on P_1 is T_4, which is then scheduled next on P_2. The process continues in this fashion, and the resulting optimal schedule is shown in Figure 8.5, where the shaded areas towards the end again indicate idle times times on the machines. The total schedule length is $C_{max} = 155$ minutes.

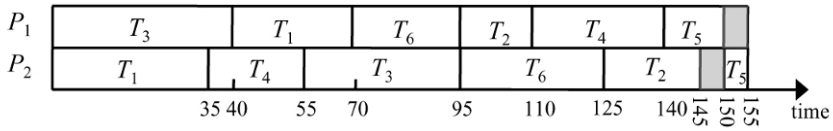


Figure 8.5

There are no simple extensions of this model that are computationally easy. The other two objectives (i.e., those that minimize mean completion time and maximal lateness) are both very difficult from a computational point of view, even for two machines. For their discussion, we refer to the advanced literature on the subject.

The second dedicated machine scheduling model is a flow shop model, i.e., a model in which each task has to be processed by all machines, but in the same, prespecified order. The schedule in Figure 8.5 does not satisfy this condition, note that for instance job T_3 is processed on P_1 first and later on P_2, while task T_1 is processed on P_2 first and then on P_1. Similar to the case of open shops, there are very few cases that are easy to solve. Among them is the case of two machines, for which the makespan is to be minimized. The solution algorithm is the famed *Johnson's rule* that was first described in the early 1950s. Assuming that all jobs have to be processed on P_1 first and then on P_2, and the processing time of task T_j is p_{1j} on machine P_1 and p_{2j} on machine P_2, the algorithm can be summarized as follows.

> *Johnson's Algorithm:* For all jobs whose processing time on P_1 is the same or less than their processing time on P_2 (i.e., $p_{1j} \leq p_{2j}$), determine the subschedule S_1 with the tasks in nondecreasing order of their p_{1j} values. For all other jobs (i.e., tasks for which $p_{1j} > p_{2j}$), determine the subschedule S_2 with all tasks in nonincreasing order of their p_{2j} values. The sequence of jobs is then (S_1, S_2).

As a numerical example for Johnson's rule, consider again Example 3, but with the proviso that each job has to be assembled first and then it is checked, i.e., it is processed on machine P_1 first before it can be processed on P_2. The tasks for which the condition $p_{1j} \leq p_{2j}$ holds are T_1, T_2, T_3, and T_6, while the jobs with $p_{1j} > p_{2j}$ are T_4 and T_5. Putting the former four tasks in nondecreasing order of the processing times on P_1 results in the subsequence $S_1 = (T_2, T_6, T_1, T_3)$, while the latter two jobs in nonincreasing order of their processing time on P_2 are put in the sequence $S_2 = (T_4, T_5)$. The resulting sequence is $(T_2, T_6, T_1, T_3, T_4, T_5)$, and the corresponding schedule is shown in the Gantt chart in Figure 8.6.

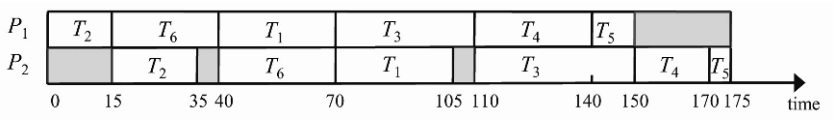


Figure 8.6

It is apparent that the schedule length is C_{max} = 175. Note the increase in the schedule length in comparison to the same example, in which the sequence of processing in the machines is not fixed (i.e., the open shop) shown in Figure 8.5. Given that a flow shop is more restrictive than an open shop (it has the additional constraint that all tasks have to be performed in the same order on the machines), the increase of the schedule length from 155 to 175 minutes is not surprising.

Simple extensions of the problem to more than two machines as well as the application of the mean flow time and the minimization of tardiness are computationally much more difficult. Again, we refer readers to the specialized literature.

The last model in this chapter deals with a job shop. Recall that by definition of a job shop, not all tasks need to be performed on all machines, and the sequence in which a job is processed on the machines is job-specific. Again, due to the inherent complexity of the problem at hand, we will restrict ourselves to the problem with two machines and the objective that minimizes the makespan. For this type of problem, Jackson described an exact algorithm in 1955. Note that Jackson's method uses Johnson's (flow shop) algorithm as a subroutine. The method can be described as follows.

Jackson's Job Shop Algorithm: Subdivide the set of jobs into four subcategories:

- J_1 includes all jobs that require processing only on machine P_1,
- J_2 includes all jobs that require processing only on machine P_2,
- J_{12} is the set of jobs that require processing on machine P_1 first and then on P_2, and
- J_{21} is the set of jobs that need to be processed on P_2 first and then on P_1.

Apply Johnson's rule to the jobs in the set J_{12}, resulting in the sequence S^{12}. Then apply Johnson's rule to the jobs in the set J_{21}, but with the processing times p_{1j} and p_{2j} exchanged. The result is the subsequence S^{21}. All jobs in the two sets J_1 and J_2 are sequenced in arbitrary order (e.g., with those jobs that have smaller subscripts scheduled first). We denote their sequences by S^1 and S^2, respectively. The jobs are then sequenced as follows: the job order on machine P_1 is (S^{12}, S^1, S^{21}), while the job order on machine P_2 is (S^{21}, S^2, S^{12}).

As a numerical illustration, we will use a modification of Example 3. The pertinent information is found in Table 8.4.

Table 8.4: Data for Example 4

Job #	T_1	T_2	T_3	T_4	T_5	T_6
Processing time p_{1j}	30	15	–	30	10	25
Processing time p_{2j}	35	20	40	–	5	30
Processing sequence	P_2, P_1	P_1, P_2	P_2	P_1	P_1, P_2	P_2, P_1

First defining the sets, we have $J_1 = \{T_4\}$, $J_2 = \{T_3\}$, $J_{12} = \{T_2, T_5\}$, and $J_{21} = \{T_1, T_6\}$. Since J_1 and J_2 include only a single task each, their respective subsequences are $S^1 = (T_4)$ and $S^2 = (T_3)$. Applying Johnson's algorithm to the tasks in the set J_{12}, we obtain the sequence $S^{12} = (T_2, T_5)$. Consider now the set J_{21}. Applying Johnson's rule to the two tasks in the set, *viz*., T_1 and T_6, with their processing times p_{1j} and p_{2j} switched, we obtain the subsequence $S^{21} = (T_1, T_6)$. Following Jackson's job shop algorithm, the overall sequence on machine P_1 is then $(S^{12}, S^1, S^{21}) = (T_2, T_5, T_4, T_1, T_6)$, while the overall sequence on machine P_2 is $(S^{21}, S^2, S^{12}) = (T_1, T_6, T_3, T_2, T_5)$. The appropriate Gantt chart is shown in Figure 8.7. It turns out that the overall schedule length $C_{max} = 130$ minutes. We also note that processor P_1 is idle for twenty minutes at the end of the schedule.

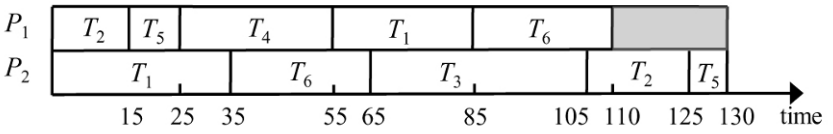


Figure 8.7

Exercises

Problem 1 (single and parallel machine scheduling): A machine scheduling problem has ten jobs with given processing times of 3, 1, 4, 1, 5, 9, 2, 6, 5, and 3 hours, respectively.

(a) Assume that the jobs are to be performed on a single machine and that the objective is to minimize mean flow time, find an optimal schedule and draw the corresponding Gantt chart.
(b) Assume that there are two parallel machine to process the jobs. Trying to minimize the schedule length, schedule the jobs using the *LPT* algorithm. Display the corresponding Gantt chart.
(c) Given two parallel machines as under (b), what is the schedule that minimizes the mean flow time? Display the corresponding Gantt chart.
(d) Reconsider questions (b) and (c) given that there are now three rather than two parallel machines.

Solution:

(a) Assuming ready times $r_j = 0$, the mean flow time is F is minimized by the shortest processing time (*SPT*) algorithm. The sequence of jobs is T_2, T_4, T_7, T_1, T_{10}, T_3, T_5, T_9, T_8, and T_6. The corresponding Gantt chart is shown in Figure 8.8.

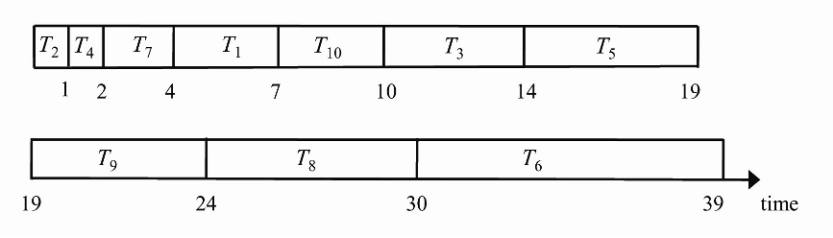


Figure 8.8

The minimal mean flow time for this schedule is then $F = \frac{1}{10}(1 + 2 + 4 + 7 + ... + 30 + 39) = 15$. Since T_1 and T_{10} both have processing times of 3 hours, they may be swapped in the optimal schedule, thus creating alternative optimal solutions. A similar argument applies to the pairs T_2 and T_4, and T_5 and T_9.

(b) With two machines P_1 and P_2, the longest processing time (*LPT*) minimizes the makespan C_{max}.

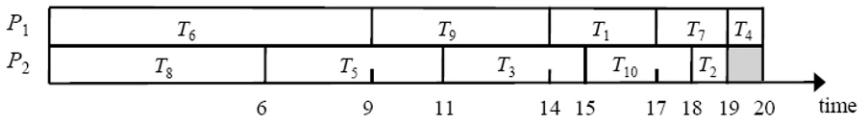


Figure 8.9

The makespan is 20 hours. In the schedule shown in Figure 8.9, one of the machines is idle for one hour.

(c) With two machines, use McNaughton's rule to minimize the mean flow time. The Gantt chart is shown in Figure 8.10.

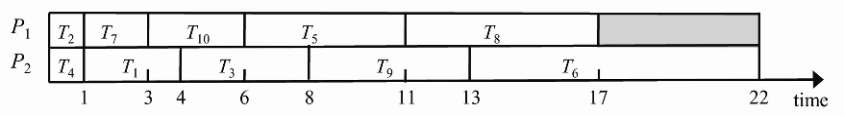


Figure 8.10

The schedule length is 22 hours and the mean flow time is $F = \frac{1}{10}(1+1+3+4+...+17+22) = 8.6$, and one processor is idle for five hours.

(d) With three machines, we use the *LPT* rule as a heuristic to minimize C_{max}. the Gantt chart is shown in Figure 8.11.

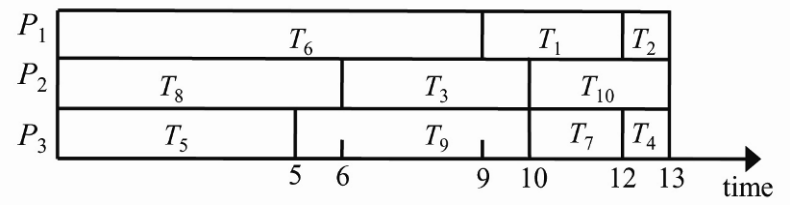


Figure 8.11

The schedule has a makespan of 13 hours. Since there is no idle time, the schedule must be optimal. Minimizing the mean flow time is done by using McNaughton's rule. The schedule is shown in Figure 8.12.

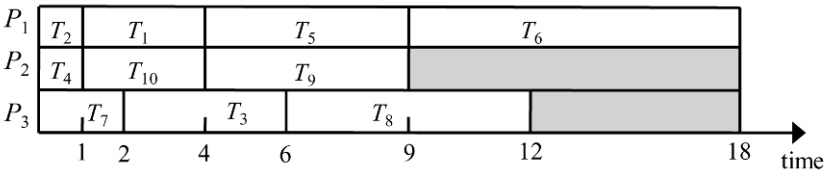


Figure 8.12

The schedule length is 18 hours and the mean flow time is $F = 6.6$. Note that there is significant idle time.

Problem 2 (open shop and flow shop scheduling): In a hospital laboratory, there are two machines testing patient blood samples. Table 8.5 shows the number of minutes required on each machine to process the samples.

Table 8.5: Data for Problem 2

Blood sample	T_1	T_2	T_3	T_4	T_5
Processing time on P_1	8	9	7	9	3
Processing time on P_2	2	3	8	4	6

(a) Assume that the order of processing the blood samples on the two machines is arbitrary. Schedule the testing on the two machines so as to minimize the schedule length. Display the optimal schedule in a Gantt chart. Indicate the schedule length as well as the idle time.
(b) Assume now that all blood samples must be processed on P_1 before they can be processed on P_2. Redo part (a) with these new assumptions.
(c) Are the optimal schedules in (a) and (b) unique? Discuss. There is no need to display Gantt charts.

Solution: (a) The *LAPT* algorithm is used to obtain the optimal schedule shown in Figure 8.13.

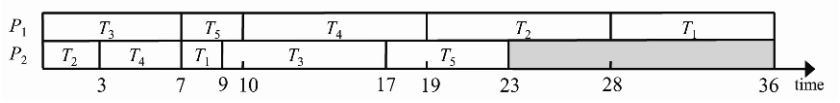


Figure 8.13

The minimal schedule length is $C_{max} = 36$ minutes. There is no idle time on P_1, while P_2 is idle at the end for 13 minutes. This is an open shop model.
(b) Johnson's rule is used to obtain the optimal schedule shown in Figure 8.14.

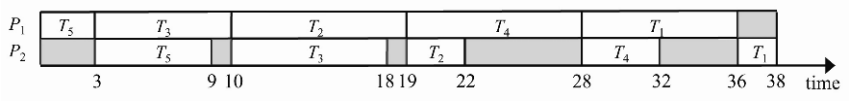


Figure 8.14

The minimal schedule length is now $C_{max} = 38$ minutes. Processor P_1 has an idle time of 2 minutes at the very end of the schedule, whereas P_2 has five separate idle time periods, totaling 15 minutes. This is a flow shop model.

(c) In (a), tasks T_2 and T_4 could swap positions in the schedule of P_1 (and on P_2 for that matter) without consequences regarding the schedule length. There are several other changes that would not destroy optimality. In (b), tasks T_2 and T_4 could also swap positions on P_1, necessitating modifications on processor P_2.

9 Decision Analysis

Everywhere in the world, at each moment, millions of people make their own decentralized decisions: when to get up in the morning, what tie to wear, what to eat for lunch or dinner, what to do in the evening (go to the theater or watch television), where to vacation, and many more. Similarly, firms will decide which mode of transportation to use when routing their products to customers, where to locate regional distribution centers, what new product lines to develop, etc. This chapter will first introduce the main elements of decision analysis, and then offer some visualizations of decision analysis problem. This is followed by a discussion of some simple decision rules, sensitivity analyses, and a discussion of the value of information. The chapter wraps up the discussion by some thoughts on utility theory.

9.1 Introduction to Decision Analysis

In order to put the decision making into a general framework, we must first distinguish between the two major elements of decision making: the *decision* made by the decision maker (whom we will think of as "us") and the *outcome* that results from our decision. Typically, the outcome is given in monetary terms, and it is usually referred to as the *payoff*.

The type of problem under consideration in this chapter is not a philosophical investigation into decisions; instead, it refers to a very specific scenario that is prevalent in decision making circumstances. In particular, we assume that there are a finite number of decisions at our disposal. Among these choices, the decision maker's task is to choose exactly one. This type of situation is often referred to as a *selection problem*. Returning to the capital budgeting decision in the introduction to integer programming in Chapter 4, we can define a binary variable y_j for each decision, such that the variable assumes a value of one if we make decision j, and 0 otherwise. A selection problem with n possible decisions will then feature the constraint $y_1 + y_2 + \ldots + y_n = 1$.

H.A. Eiselt and C.-L. Sandblom, *Operations Research: A Model-Based Approach*,
DOI 10.1007/978-3-642-10326-1_9, © Springer-Verlag Berlin Heidelberg 2010

However, in contrast to problems that can simply be formulated as integer programming problems, there are two possible extensions that typically arise. The first extension involves the evaluation of a decision on more than a single criterion. As an example, consider a department store that considers changing the layout of its store. There are, of course, the costs of such a decision. But there is more: there is the changed customer flow that may result in higher exposure of some goods to customers and resulting potential higher sales of these products (the main reason for a change in layout), the (temporary) confusion of customers who may refrain from purchasing at the store, the potential retraining of employees if the layout change was in conjunction with new products added to the goods available at the store, and so forth. Similarly, consider the construction of a new office highrise building in an urban area. Concerns will include costs, safety (e.g., evacuation routes in case of emergencies), reimbursement of nearby apartment renters for the loss of view and sunlight that may be blocked by the new building, parking for employees and customers, and many more. Selection problems in which multiple criteria are considered are called *multicriteria decision making* problems (or *MCDM*). On the other hand, suppose that we only consider a single criterion, but the outcome of our decision is no longer certain. This is the standard scenario in *decision analysis* or, as it is frequently called, *games against nature*. The name stems from game theory and can be explained as follows. Consider a standard two-person-game with two players, one decision maker (us), and the other being our opponent (nature). Each of the two players has a number of possible actions at his disposal: the decision maker has the decisions, while nature controls the "states of nature," nature's equivalent of the decision maker's decision choices. This is the scenario examined in this chapter.

There is, however, a fundamental asymmetry in games against nature. First of all, the combination of the decision maker's choice and nature's state of nature will determine the outcome for the decision maker (nature will face no outcome). Secondly, the decision maker will examine the possible outcomes of his decisions before choosing one, while nature does not consider the outcomes, but chooses her strategies according to some probability distribution. This is the reason why the decision maker is usually called "intelligent" (we prefer to think of it as "rational"), while nature is referred as a "random player."

As an illustration, consider the following numerical example with three decisions d_1, d_2, and d_3, and four states of nature s_1, s_2, s_3, and s_4. The payoffs are shown in Table 9.1.

Table 9.1: Payoffs

	s_1	s_2	s_3	s_4
d_1	3	–2	4	6
d_2	2	0	–4	1
d_3	5	2	0	–3

The decision maker could argue that d_1 is best, as under three states of nature there is a reasonable payoff, while its largest possible loss is –2 and as such not as bad as the losses that can occur with the other two decisions. If the decision maker were to choose d_1, while nature would randomly choose s_3, then the decision maker would obtain a payoff of 4.

It is important to realize that nature's decision is either made simultaneously with that of the decision maker without cooperation, or, equivalently, the decision maker chooses first, followed by nature's choice. As in all game-theoretic situations, it is crucial to specify which player knows what and when. This also marks the distinction concerning the level of certainty the decision maker has about nature's choice. As usual, consider the extremes first. If the decision maker knows with certainty what nature is going to do, we face a decision problem under certainty. Given the scenario of selection problems, this means that the decision maker knows which column of the payoff matrix will apply. In the example of Table 9.1, suppose that the decision maker knows that nature will choose s_1. This means that the consequences of the decisions are known with certainty: choosing d_1 will result in a payoff of 3, choosing d_2 will result in a payoff of 2, and choosing decision d_3 will yield a payoff of 5. Clearly, the decision maker's payoff is maximized by the payoff of 5, which we arrive at by choosing d_3. This is the optimal solution and the problem is solved. (Before we continue with different levels of knowledge, note that the decision maker has jurisdiction only over his choices, e.g., d_3, but he cannot choose the payoff directly).

On the other extreme is *uncertainty*. In decision making under uncertainty, the decision maker has absolutely no idea about nature's choice. Uncertainty is quite rare, it may occur in the performance of new and untried products, behavior of customers in new and untested markets, and other situations, in which no information is provided. As in all decision-making situations, if the level of input into the problem is low, the output will have to make do with simplistic rules. This is precisely what happens in this situation as will be seen below.

Clearly, there is much territory between certainty and uncertainty. One milestone in between is decision making under *risk*. In decision making under risk, the decision maker is assumed to know the probability distribution. Examples would include past weather observations for farmers, predictions concerning customer behavior based on similar situations, and so forth. Rules for decision making under risk are described below.

9.2 Visualizations of Decision Problems

Before getting into specific rules for different situations, we would like to describe some ways to visualize decision making scenarios. Different visualization on different levels are available. For the macro view, there are *influence diagrams*. The idea is to show the basic interdependencies between decision and outcome,

while ignoring details. Their biggest strength is clarity, which is achieved by their concentration of fundamental relations and their resulting small size. On the other hand, *decision trees* are used for the micro view. They include specific decision rules at each step of the way. Consequently, they tend to be large and cumbersome to deal with.

In order to demonstrate influence diagrams, we first make a list with three columns. The first column includes the decision maker's possible decisions, the second column represents the random events that somehow influence the decisions and the outcomes, and the last column includes the consequences that are a result of the decisions and the random events. As an illustration, consider a department store that contemplates adding an electronics department to its services. In case the introduction is accepted by the public, management considers relocating that department into a separate building. The aforementioned listing is shown in Table 9.2.

Table 9.2: Decision, Events, and Consequences for Influence Diagram

Decision	Random event	Consequence
Add electronics department	General economic conditions	Profit
Relocate department into a separate building	Local acceptance of services	

In order to visualize the problem, we can create an influence diagram, in which our decisions are shown by rectangles, random events are shown as circles, and consequences are shown by triangles. We then add directed arcs, so that an arc from some node i to another node j exists, if it is believed that i influences j. The influence diagram for our problem could look as shown in Figure 9.1.

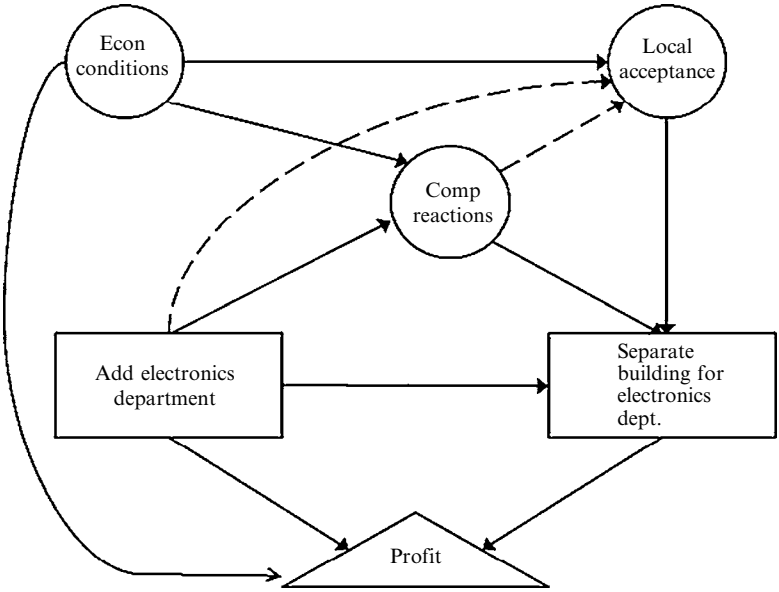


Figure 9.1

The broken arcs in the figure are somewhat tenuous: they indicate the belief that local acceptance of an electronics department or store is influenced by the existence of an electronics department in our department store and our competitors' reaction to our introduction of the department.

On the other hand, we could zoom in and outline our decisions, our competitors' decisions, and economic conditions in a decision tree. Decision trees have decisions and events listed next to arcs. A square node indicates that we will make a decision, a round node means that a decision is made not within our control (i.e., a random event), and a triangular node denotes the end of this branch of the tree. Normally, there will be an indication next to a triangular node what the payoff is to us at that point. For now, we will concentrate on the structure of the tree and return to the numerical aspects of decision trees in a detailed discussion in Section 9.5 of this chapter. A possible decision tree for our sample problem is shown in Figure 9.2.

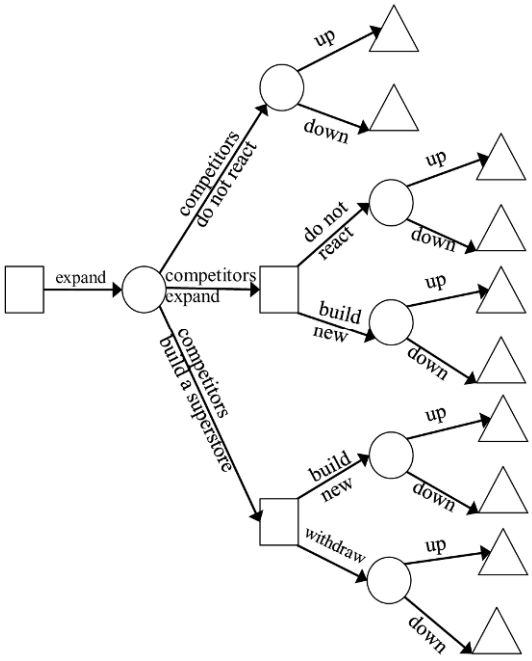


Figure 9.2

It indicates that after we have decided to expand, our competitor(s) could either not react, also expand their stores accordingly, or build a superstore. While these are not random events, they are shown here as such as these decisions are not within our control. In case our competitors do not react, our profit will depend on the state of the economy, which is shown in the Figure simply as "up" or "down." (This exceedingly simplistic notion has been used so as to save space—it should have become clear by now that decision trees tend to get very large even if the players do not have very many options). In case our competitor(s) expand, we can either do not react ourselves or build the planned new building. Each of our decisions will be followed by a random event, at which nature decides which turn the economy takes. Finally, if the competitors have decided to build a superstore, we may either withdraw due to limited options to raise capital, or build the planned addition. Again, each decision is followed by the economy going up or down.

9.3 Decision Rules Under Uncertainty and Risk

Our discussion below will be based on a numerical example with the payoff matrix

	s_1	s_2	s_3
d_1	2	−2	5
d_2	0	−1	7
d_3	2	1	1
d_4	2	−3	4

Before performing any calculations, it is useful to first examine the payoff matrix regarding *dominances*. A decision i (meaning row) dominates a decision k (row) if all elements in row i are at least as large as the corresponding payoffs in row k. In other words, for each state of nature, decision i is at least as good as decision k. If this is the case, then decision k can be deleted from consideration. The determination whether or not dominances exist in a problem requires pairwise comparisons. In our example, let us first compare decisions d_1 and d_2. Given the first state of nature s_1, decision d_1 results in a payoff of 2, while d_2 nets us only 0, so d_1 is preferred. However, given s_2, decision d_1 results in a loss of 2, while d_2 results in a loss of only 1, so that d_2 is preferred. This means that neither decision is consistently better than the other. Comparing d_2 and d_3 also results in no dominance (d_3 is preferred in case s_1 or s_2 occurs), while d_2 is preferred to d_3 in case s_3 comes up. However, the picture changes when comparing d_1 and d_4. Here, it is apparent that d_1 and d_4 are equally good given s_1, while d_1 is better than d_4 in case of s_2 and s_3, so that d_1 dominates d_4; thus d_4 can be deleted. The examination would have to continue comparing d_1 and d_3.

Given m decisions, $\frac{1}{2}m(m-1)$ pairs of decisions will have to be compared. If any dominances are missed by accident, no harm is done: the model will be a bit bigger than it has to be, but no "reasonable" rule will choose a dominated decision. Note that we cannot apply the concept of dominance to the states of nature (i.e., the columns of the payoff matrix). The reason is that the concept of dominances is based on a comparison of the payoffs, i.e., a rational decision maker, and nature is no such player.

Consider now a decision making problem under uncertainty. One simple decision rule will attempt to guard against the worst case. It has a variety of names, *Wald's rule* named after its inventor, the *pessimistic rule* based on the mindset of the decision maker, and the *maximin rule* based on the way the rule works. First of all, we will determine the *anticipated payoffs* associated with our decisions. In our example, a pessimist choosing d_1 would assume that the worst case applies and the payoff will be −2. Similarly, decision d_2 would result in a payoff of −1, and so forth. The vector of anticipated payoffs would then be $\mathbf{a} = [-2, -1, 1, -3]^T$. Formally, these are the row minima of the payoff matrix. Choosing the best among these decisions will then be d_3 as it leads to the maximum payoff among the anticipated payoffs. This is the reason for calling the rule a "maximin" rule.

While the rule surely protects against the worst case, it has a number of shortcomings. The most predominant problem with it is its exclusive focus on the worst case. For instance, if someone had to pick up his multimillion dollar winnings from the

lottery office, the worst-case rule would suggest that he not do that: while walking or driving to the office, he might get hit by a truck and die, the worst case that has to be avoided. Since it is unknown how likely (or, in this case, how unlikely) such an incident would be, a decision made on the basis of Wald's rule would try to prevent it.

Another, very similar, rule is the *optimist's rule*. An optimist's anticipated payoff would include the best possible payoff for each decision, i.e., the row maxima. In our example they would be $\mathbf{a} = [5, 7, 2, 4]$. The optimist would then choose the decision with the highest anticipated payoff, leading to a *maximax rule*. In our illustration, d_2 would be the optimist's choice. This Polyanna-inspired rule suffers from the same limitations as Wald's rule, except that it has replaced guarding against the worst case by anticipation of the best case.

A third rule for making decisions under uncertainty was independently developed by Savage and Niehans. It is called the *minimax regret criterion* and it has become the basis of what is often referred to as *robust optimization*. The idea is to compare for each state of nature the payoff the decision maker gains with a decision and the best possible payoff that could have been obtained given the same state of nature. In our numerical example, we compute the regret of decision d_2 given the second state of nature s_2. The payoff to the decision maker is -1. However, if the decision maker had just known in advance which state of nature would occur (the second), he could have his best response d_3, which would have led to a payoff of 1, the highest payoff given s_2. The difference between the actual payoff and the best possible payoff under that state of nature gives a regret of $1 - (-1) = 2$. These regrets are calculated for all pairs of decisions and states of nature, and they form the *regret matrix*. In our example, the regret matrix is

$$\mathbf{R} = \begin{bmatrix} 0 & 3 & 2 \\ 2 & 2 & 0 \\ 0 & 0 & 6 \\ 0 & 4 & 3 \end{bmatrix}$$

Note that regrets are never negative. Furthermore, there is always at least one zero in each column of the regret matrix (belonging to the element that determines the column maximum). The decision maker can then apply any rule on the regret matrix rather than the original payoff matrix. If we were to use the pessimist's rule onto the regret matrix $\mathbf{R}$, we would anticipate regrets of $\mathbf{r} = [3, 2, 6, 4]$. Note that these are the row maxima, not the row minima as used above. The reason is that a payoff matrix includes payoffs that the decision maker would like to maximize, whereas the regret matrix features regrets that, similar to costs, the decision maker would like to minimize. Among the anticipated regrets, the decision maker will then choose the lowest, which in our example is d_2, a decision leading to an anticipated regret of 2. Applied to the regret matrix, Wald's rule is a minimax rule in contrast to the maximin version that is used in case a payoff matrix is given.

Consider now decision making under risk. In this scenario, we can associate with each state of nature a probability p_j, which has been determined by past observations. *Bayes's rule* is then used to compute the expected values and choose the decision that leads to the maximum expected payoff, making Bayes's criterion a *weighted maxisum rule*. In our example, suppose that the probabilities of the three states of nature have been determined as $\mathbf{p} = [.5, .3, .2]$. The *expected payoffs* (or *expected monetary values EMV*) are then as follows:

$$EMV(d_1) = 2(.5) - 2(.3) + 5(.2) = 1.4,$$
$$EMV(d_2) = 0(.5) - 1(.3) + 7(.2) = 1.1,$$
$$EMV(d_3) = 2(.5) + 1(.3) + 1(.2) = 1.5, \text{ and}$$
$$EMV(d_4) = 2(.5) - 3(.3) + 4(.2) = .9.$$

Thus the anticipated payoffs are 1.4, 1.1, 1.5, and 0.9, so that the decision maker will choose d_3, a decision that has the highest expected payoff of 1.5. In case of a tie, it is possible to use secondary criteria.

Bayes's rule has been popularized by what is known as the *Newspaper Boy Problem*. The decision of the newspaper boy concerns the number of newspapers he will purchase in the presence of demand uncertainty. The probabilities of the demand are based on past experience. If the boy buys too many papers on a slow day, he will have papers left over, which will have to be disposed of for their very low salvage value. On the other hand, if he purchases to few and the paper turns out to have some interesting stories, he will not have enough papers to sell, so that not only does he lose business today, but may irritate customers who may purchase their papers elsewhere in the future.

As an illustration, consider the following numerical

Example: A newspaper boy knows that he can sell either 10, 20, 30, or 40 newspaper on any given day (barring days with major headlines, such as assassinations, wars, or the latest replacements of body parts of some actress). The boy will purchase a newspaper for 20¢ and they sell for 90¢. The salvage value of an unsold newspaper is 5¢, while the opportunity cost for newspapers has been estimated to be 15¢ for each newspaper that could have been sold but was not due to the lack of supply. The payoff matrix for the newsboy problem is then

$$\mathbf{A} = \begin{array}{c} \\ d_1 \\ d_2 \\ d_3 \\ d_4 \end{array} \begin{array}{c} \begin{array}{cccc} s_1 & s_2 & s_3 & s_4 \end{array} \\ \begin{bmatrix} \$7.00 & 5.50 & 4.00 & 2.50 \\ 5.50 & 14.00 & 12.50 & 11.00 \\ 4.00 & 12.50 & 21.00 & 19.50 \\ 2.50 & 11.00 & 19.50 & 28.00 \end{bmatrix} \end{array}$$

where the decisions d_1, d_2, d_3, and d_4 refer to the newspaper boy buying 10, 20, 30, and 40 newspapers, while the states of nature s_1, s_2, s_3, and s_4 refer to a demand of 10, 20, 30, and 40, respectively. Note that all entries on the main diagonal refer to cases, in which the sale equals the number of newspapers that were purchased, while all entries above the main diagonal involve some unsatisfied demand with its opportunity costs, while the entries below the main diagonal involve a surplus of newspapers, so that salvage values have to be applied. Given probabilities of $\mathbf{p} = [.6, .2, .1, .1]$ for the four states of nature, the four decisions have expected payoffs of \$5.95, \$8.45, \$8.95, and \$8.45, so that the newspaper boy will buy 30 newspapers and expect a daily payoff of \$8.95.

Applications of this type occur in many circumstances, in which we are dealing with perishable goods. An excellent example is the airline industry. Seats in an airplane are a perishable commodity, as, once the airplane door closes, an empty seat is worthless. The airline's decision problem is then to choose among their aircraft the type that best fits the expected demand on each of the routes.

Back to our discussion of different approaches to decision making. In decision making under risk, there is also the possibility to use *target values*. The idea with this approach is to choose the decision that provides the highest probability that the payoff does not fall short of a predetermined target value. To illustrate, use again the example introduced earlier in this section. Suppose that a target value $T = 1$ has been chosen. Decision d_1 will then achieve this target value only if nature chooses either s_1 or s_3 as her strategy, which will happen with a probability of 0.5 and 0.2, respectively. This means that when using d_1, there is a probability of 0.7 that the target value is reached or exceeded.

When using decision d_2, the target value $T = 1$ will be achieved only if s_3 comes up, which happens with a probability of 0.2. Similarly, decision d_3 reaches the target value in case nature plays s_2 or s_3, so that the probability of a payoff of at least T equals 0.5, and for d_4 the probability is 0.7. The decision maker will then choose the decision that maximizes the probability of getting at least $T = 1$, which is done by choosing either d_1 or d_4. Note that one of those optimal choices is the dominated decision d_4. This is possible; however, a dominated decision can never be the unique optimum for any "reasonable" decision rule.

In order to derive a simple decision tool, we plot the probability that a decision achieves at least a prespecified target value T against the full range of target values T. For our example, Figure 9.3 provides the graph in question. In particular, the solid line shows the achievements of decision d_1, the broken line is for d_2, and the dotted line shows the results for d_3. For clarity, we ignore the dominated decision d_4.

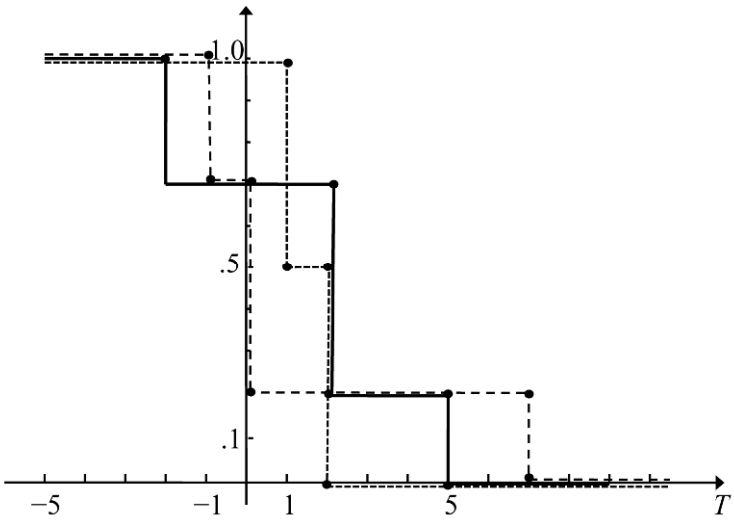


Figure 9.3

Given the graph in Figure 9.3, we can very simply determine which decision has the highest probability of reaching a target value. For instance, if the target value were $T = -1.5$, then decisions d_2 and d_3 both have a probability of "1" to achieve this value, while decision d_1 has only a probability of .7 of achieving this payoff. In other words, we are interested in the "upper envelope," i.e., the highest of all of the functions. Given that, we can determine which function is highest and summarized it in the following decision rule:

If $T < -2$, any decision will achieve the target.
If $T \in [-2, -1]$, d_2 and d_3 are best. Both will reach the target with a probability of 1.
If $T \in [-1, 1]$, d_3 is best. It reaches the target with a probability of 1.
If $T \in [1, 2]$, d_1 is best. It reaches the target with a probability of .7.
If $T \in [2, 5]$, d_1 and d_2 are best. Both achieve the target with a probability of 0.2.
If $T \in [5, 7]$, d_2 is best. It achieves the target with a probability of .2.
If $T > 7$, none of the decisions will be able to reach the target.

A loose summary will indicate that d_3 is best for low target values, d_1 is best for intermediate target values, while d_2 is best for high target values. This is, of course, not surprising: decision d_3 has no extremes on the low end, while d_2 does have an extreme possible payoff on the high end.

9.4 Sensitivity Analyses

This section will examine two types of sensitivity analyses. The first (simpler) case assumes that a payoff, i.e., one element of the payoff matrix, is no longer known with certainty. The idea is to develop simple decision rules that provide guidance to the decision maker in case the payoff changes. Again, we will base our arguments on the example introduced at the beginning of this chapter, which is shown again here for convenience.

	s_1	s_2	s_3
d_1	2	–2	5
d_2	0	–1	7
d_3	2	1	1
d_4	2	–3	4
p	.5	.3	.2

Suppose now that there is some uncertainty concerning the payoff of the second decision in case of the third state of nature. We can then write the payoff as $a_{23} = 7 + \varepsilon$ with an unknown $\varepsilon \in [-2, 3]$. In other words, we expect the actual payoff to be somewhere between 5 and 10. The expected payoffs can then be computed as

$$EMV = \begin{bmatrix} 1.4 \\ 1.1 + .2\varepsilon \\ 1.5 \\ .9 \end{bmatrix}.$$

Here, we can ignore decisions d_1 and d_4 as, regardless of the value of ε, decision d_3 is better than those. This leaves us with the comparison between d_2 and d_3. Figure 9.4 plots the expected monetary values of the two decisions as a function of the change ε.

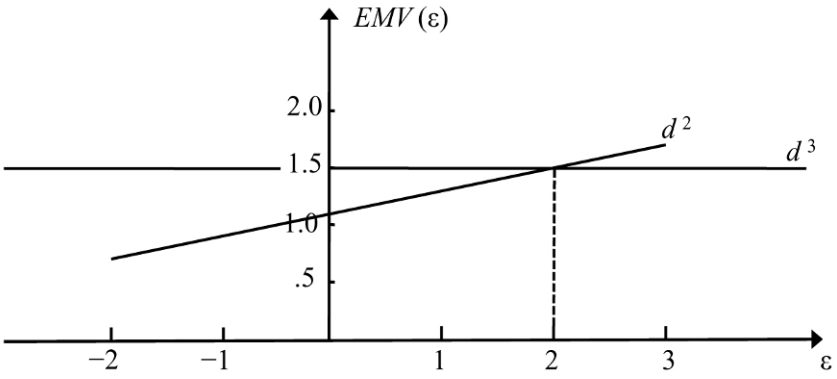


Figure 9.4

The two payoff curves $EMV(d_2)$ and $EMV(d_3)$ intersect at $\varepsilon = 2$. To the left of $\varepsilon = 2$, the payoff is higher for decision d_3, while to the right of $\varepsilon = 2$, the payoff with d_2 is higher. This leads us to the following decision rule:

> If $\varepsilon \geq 2$ (or, alternatively, $a_{23} \geq 9$), then decision d_2 is best, and
> if $\varepsilon \leq 2$ (or, alternatively, $a_{23} \leq 9$), decision d_3 is best.

Next, consider the possibility that there is some uncertainty surrounding a probability estimate. This case is somewhat more difficult conceptually, as the increase or decrease of a single probability will necessarily imply that other probabilities change as well, based on the simple fact that the sum of probabilities equals one. Back in our numerical example with the original payoffs, we are now no longer certain about the probability of s_1.

We may assume that an increase of p_1 by some unknown value ε may reduce the probabilities of all other states of nature by the same amount, and similar for a decrease of p_1. If this assumption were reasonable, we then have probabilities $[p_1 + \varepsilon, p_2 - \tfrac{1}{2}\varepsilon, p_3 - \tfrac{1}{2}\varepsilon]$, or, with the values of p_1, p_2, and p_3, we have $[.5 + \varepsilon, .3 - \tfrac{1}{2}\varepsilon, .2 - \tfrac{1}{2}\varepsilon]$. Given these probabilities, we can then again compute the expected monetary values, which are

$$EMV(\varepsilon) = \begin{bmatrix} 1.4 + .5\varepsilon \\ 1.1 - 3\varepsilon \\ 1.5 + 1\varepsilon \\ .9 + 1.5\varepsilon \end{bmatrix}$$

Suppose now that it has been estimated that p_1 will assume a value somewhere between .3 and .6. In other words, starting with its present value of $p_1 = .5$, the change $\varepsilon \in [-.2, +.1]$. We can now again plot the expected monetary values of the decisions as functions of ε. This is shown in Figure 9.5.

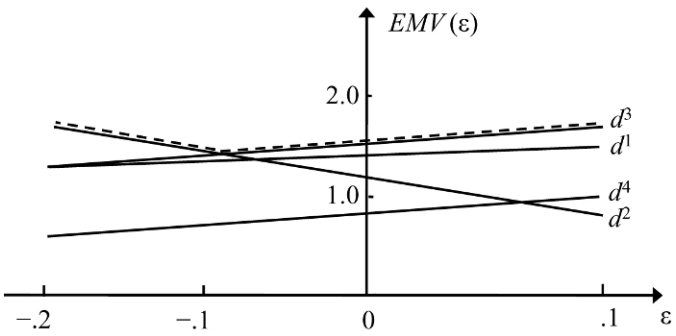


Figure 9.5

For any value of ε we are interested in the highest expected payoff, i.e., in the point on the highest curve (something called the upper envelope, shown here by the broken line). We observe in Figure 9.5 that the decisions d_1 and d_4 are dominated by d_3, leaving d_2 and d_3 as the only decisions of interest. The two functions intersect where their payoffs are equal, i.e., at the point at which $1.1 - 3\varepsilon = 1.5 + 1\varepsilon$. Solving for ε, we obtain $\varepsilon = -.1$ This leads to the following decision rule:

> If $\varepsilon \le -.1$ (or, alternatively, $p_1 \le .4$), then decision d_2 is best, and
> if $\varepsilon \ge -.1$ (or, alternatively, $p_1 \ge .4$), then decision d_3 is best.

Note that it is not generally true that decision rules such as this consist of only two parts. It is possible that any number of the existing decisions may be best for some range of changes.

It is, of course, possible that a change of p_1 does not affect the remaining probabilities equally. For instance, it could be estimated that an increase of p_1 by some unknown value ε will decrease p_2 by ⅔ε, while p_3 will decrease by ⅓ε. The expected payoffs are then

$$\begin{bmatrix} 1.4+\frac{5}{3}\varepsilon \\ 1.1-\frac{5}{3}\varepsilon \\ 1.5+\varepsilon \\ .9+\frac{8}{3}\varepsilon \end{bmatrix}.$$

The payoff functions (as functions of ε) are shown in Figure 9.6.

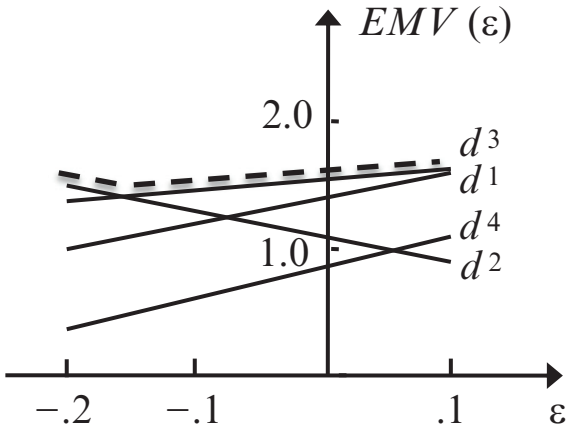


Figure 9.6

As above, we assume that $\varepsilon \in [-.2, +.1]$. A similar analysis to that provided above reveals that decision d_3 dominates d_1 and d_4, leaving the decision maker with d_2 and d_3. Again, we are interested in the upper envelope, shown here by the broken line. The expected monetary values of the two decisions are equal if $1.1 - 5/3\varepsilon = 1.5 + \varepsilon$, i.e., for $\varepsilon = -.15$. This leads to the following decision rule:

> If $\varepsilon \leq -.15$ (or, equivalently, $p_1 \leq .35$), then decision d_2 is optimal, while if $\varepsilon \geq .15$ (or, equivalently, $p_1 \geq .35$), then decision d_3 is optimal.

9.5 Decision Trees and the Value of Information

In this section we will again consider decision making problems under risk. In addition to the decision rules discussed in the previous section, we will determine the value of information that goes beyond the probabilities for the states of nature that we continue to assume to be known. We will commence our discussion with an extreme case known as the *expected value of perfect information* (*EVPI*). Clearly, no information is perfect, but this value provides an upper bound for the value of any information, as no information can be worth more than perfect information. Since it is easy to compute, it provides the decision maker with a ballpark figure. In simple words, the *EVPI* is the difference of the payoff with perfect information and the best we can do without any information beyond what is included in the standard setting. As an illustration, consider again our example. Recall that with the probabilities of 0.5, 0.3, and 0.2 of the three states of nature, the highest expected monetary value was $EMV^* = 1.5$, which was achieved by choosing decision d_3, where we use an asterisk to indicate optimality. This is the best the decision maker can do without additional information. Consider now perfect information. It means that the decision maker will know in advance which state of nature will occur. It is important to realize that this does *not* mean that the decision maker can change the probabilities of the states of nature—all we assume that the decision maker knows which state of nature occurs before he makes his own decision.

In our numerical example, the best response to the first state of nature s_1 is to use d_1, d_3, or d_4; each of these responses will result in a payoff of 2 to the decision maker. Similarly, if the decision maker knows that s_2 occurs, his best response is to choose d_3, which results in a payoff of 1. Finally, if nature chooses s_3 and the decision maker knows about it beforehand, the best response is d_2, netting 7. The payoff matrix **A** is shown again below with the starred element indicating those payoffs that result from the decision makes best response to nature's action.

$$\mathbf{A} = \begin{bmatrix} 2* & -2 & 5 \\ 0 & -1 & 7* \\ 2* & 1* & 1 \\ 2* & -3 & 4 \end{bmatrix}$$

Given the known probability distribution **p** = [.5, .3, .2], we can state that 50% of the time, nature chooses s_1 and the decision maker obtains a payoff of 2 (the first column), 30% of the time, nature chooses s_2 and the decision maker's best reaction results in a payoff of 1, and finally, 10% of the time nature chooses s_3 and the decision maker's response is d_2, resulting in a payoff of 7. Hence, the *expected payoff with perfect information* is $EPPI = 2(.5) + 1(.3) + 7(.2) = 2.7$. The expected value of perfect information is then $EVPI = EPPI - EMV^* = 2.7 - 1.5 = 1.2$. As indicated above, this is an upper bound on the amount of money that the decision maker should be prepared for any type of additional information.

Consider now a situation, in which the information provided to the decision maker is imperfect. Imperfect information is usually provided by indicators. As an example, consider the price of an individual stock. It is usually not possible to get any direct information, so that we must rely on a proxy, such as demand for products of firms in that industry or manufacturers' receipts. Clearly, such proxies are only of value if there is a links between them and the state of nature we want to forecast. For instance, it would be meaningless to forecast the probability of sales of a new camera by using the demand for potatoes. The stronger the links between a proxy and the state of nature it is used to forecast, the closer we will be to perfect information. The strength of the link between indicator and state of nature is typically provided by a table of conditional probabilities. For our example, assume there are two indicators I_1 and I_2, whose links to the three states of nature is shown in Table 9.3.

Table 9.3: Conditional probabilities $P(I|s)$

	s_1	s_2	s_3
I_1	.6	.9	.2
I_2	.4	.1	.8

In other words, given that the first state of nature s_1 will eventually occur, indicator I_1 has a 60% chance of coming up, and similar for the other values. It is apparent that the probabilities in each column add to one. (In the extreme case, there would be three indicators and the probabilities $P(I_1|s_1) = P(I_2|s_2) = P(I_3|s_3) = 1$ and all other conditional probabilities equal to zero. Then the three indicators are sure predictors of the states of nature, and we have again perfect information. This is the limiting case).

Before performing any computation, we will first depict the structure of the decision-making process in the form of a *decision tree*. The general structure of such a tree is shown in Figure 9.7.

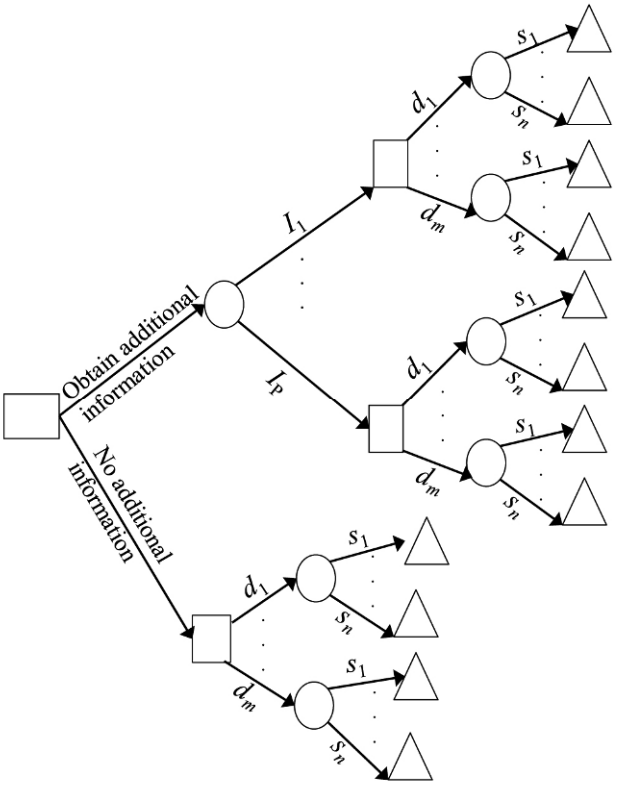


Figure 9.7

In Figure 9.7, we start on the left with the square node (the root of the tree), indicating that we have to make a decision first. Our decision is whether or not to solicit additional information. The lower branch shows that no additional indication is sought, and it will end up with what we have already done earlier when applying Bayes's rule. Often, this part of the tree is deleted and its outcome at the end of the first branch is shown as *EMV** (= 1.5 in our example).

Consider now the upper branch that indicates that we ask for additional information. This will be followed by a random event according to which some indicator comes up. this is shown by the circular node, followed by branches for all indicators I_1, …, I_p. Once we have received an indicator (meaning we now have additional information), we must make one of our decisions d_1, …, d_m. This is shown again as a decision node, followed by arcs, one for each of our decisions. Finally, once

we have made a decision, a random event, i.e., one of the states of nature will occur. The end of the sequence of decisions and events is marked by a triangle, next to which we will place the outcome for that particular scenario. It is important to realize that each endpoint marked with a triangle actually symbolizes a scenario that includes the entire sequence of decisions and events from the root of the tree to that triangle. For instance, the topmost triangle on the right of the tree indicates that we asked for additional information, received the indicator I_1, made decision d_1, and then state of nature s_1 occurred.

Once the structure of the tree has been determined, we need to put some numbers into the tree. As already mentioned above, the payoffs are taken directly from the payoff matrix and put next to the triangles on the right side of the decision tree. What are now needed are probabilities. More specifically, we need two types of probabilities. The first types are associated with the random events that govern which of the indicators comes up. They are called indicator probabilities $P(I_k)$. So far, we do not have these probabilities. The second type of probabilities are associated with the states of nature occurring at the very end of the decision tree, just before the payoffs are due. At first glance, it would appear that the probabilities that we used in Bayes's rule (so-called *prior probabilities*) $P(s)$ should be used. This is, however not the case. The reason is that a state of nature occurs after an indicator has come up. And the whole point of indicators is that they are not independent from the states of nature.

So what we need are the so-called *posterior probabilities* $P(s|I)$. In other words, these are conditional probabilities that specify the likelihood that a state of nature occurs, given that an indicator has come up earlier. And while it may appear that we are given the posterior probabilities in a table such as Table 9.3, this is not the case: while both are conditional probabilities, Table 9.3 includes probabilities of the type $P(I|s)$, posterior probabilities are $P(s|I)$. In other words, the probabilities will have to be inverted, which is done by what is known as *Bayes's theorem*. This theorem (or rather conversion rule) is explained in Appendix D of this book. We will used it here in a very convenient computational scheme shown below. And a byproduct of the conversion are the indicator probabilities that we also need to put numbers on our decision tree.

The computational scheme that determines indicator probabilities and posterior probabilities deals with each of the indicators separately. Consider again our numerical example and use only the first indicator I_1. Table 9.4 shows the computational scheme we use as it applies to the indicator I_1. The first column lists the states of nature, and the second column includes their prior probabilities. The third column includes the conditional variables that relate to the indicator under consideration (here I_1) and all states of nature. In other words, the third column is nothing but the first row in Table 9.3. Following Bayes's rule, we then multiply the elements in the second and third columns and put them in the fourth column. Their sum, shown at the bottom of column four, is then the indicator probability for the indicator under consideration, here $P(I_1)$. Finally, the posterior

probabilities in the last column are obtained by dividing each element of column four by the indicator probability, i.e., $P(s_i|I_1) = P(I_1|s_i)P(s_i)/P(I_1)$.

Table 9.4: Computation of $P(I_1)$ and $P(s|I_1)$

s	$P(s)$	$P(I_1\|s)$	$P(I_1\|s)P(s)$	$P(s\|I_1)$
s_1	.5	.6	.30	.4918
s_2	.3	.9	.27	.4426
s_3	.2	.2	.04	.0656
			$P(I_1) = .61$	

This procedure provides us with both, the indicator probabilities and the posterior probabilities needed to complete the numerical information required in the decision tree. Similar computations are then performed for the second indicator. The results are shown in Table 9.5. Note that the sum of indicator probabilities must equal one. The same applies to the posterior probabilities in the rightmost columns of Tables 9.4 and 9.5.

Table 9.5: Computation of $P(I_2)$ and $P(s|I_2)$

s	$P(s)$	$P(I_2\|s)$	$P(I_2\|s)P(s)$	$P(s\|I_2)$
s_1	.5	.4	.20	.5128
s_2	.3	.1	.03	.0769
s_3	.2	.8	.16	.4103
			$P(I_2) = .39$	

The decision tree with all numerical information is then shown in Figure 9.8.

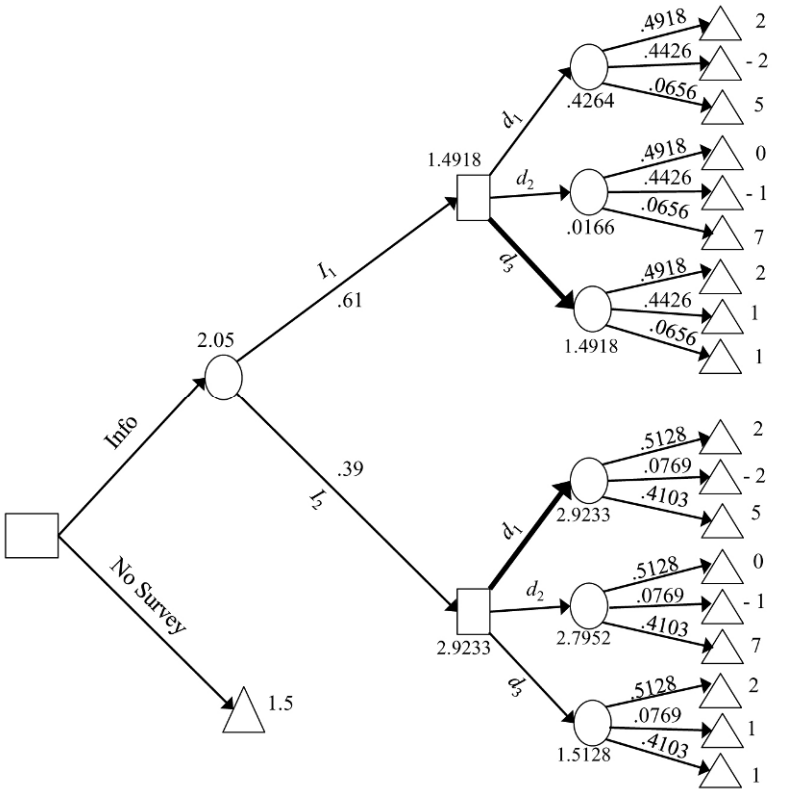


Figure 9.8

At this point, all numerical information is available and we need to discuss a technique that determines the *EPSI*.

The general idea to deal with decision trees is to use a recursive procedure. This procedure starts from the leaves of the tree and works back to the root. There are two rules in doing so:

Rule 1: Moving back into an event node, we take expected values, and
Rule 2: Moving back into a decision node, we choose the decision with the highest expected payoff.

Applying the two rules to our example, we start in the northeast corner of Figure 9.8. Following the top three branches with the payoffs of 2, –2, and 5 at their respective ends, backwards, we reach an event node. This means that we use the posterior probabilities of the branches to calculate expected payoffs at the event node at the beginning of the branches. In this case, we compute 2(.4918) – 2(.4426) + 5(.0656) = .4264. A similar process is used for the other branches.

The next step starts from the (total of six) event nodes whose expected payoffs have just been computed. Starting again from the top with the three event nodes labeled with .4264, .0166, and 1.4918, we will follow the arcs that lead into these nodes one step backwards. Since these arcs lead into a decision node, we will choose the decision that results in the highest expected payoff. In this case, the highest payoff is 1.4918 and the decision that leads to this payoff is d_3. Similarly, comparing the expected payoff in the lower part of the diagram (2.9233, 2.7952, and 1.5128), the highest expected payoff is 2.9233 and the decision leading to this payoff is d_1. The arcs of the two decisions chosen in this step are bolded.

We now have two decision nodes labeled with 1.4918 and 2.9233, respectively. Moving back another step means going into an event node, which is done by computing the expected payoff by using the indicator probabilities 0.61 and 0.39. This results in the expected payoff with imperfect information $EPII = 2.05$. Similar to the definition of the expected value of perfect information (the difference between the expected payoff with and without this information), we can now define the *expected value of imperfect information*, most frequently called *expected value of sample information EVSI*. The *EVSI* is defined as the difference between the expected payoff to the decision maker with and without sample information. Formally, we obtain $EVSI = EPII - EMV^* = 2.05 - 1.5 = 0.55$. This is the highest amount that we should be prepared to pay for this information.

The final step consists of the computation of the *efficiency E*. The efficiency measures how close the sample information is in comparison with perfect information, i.e., $E = EVSI/EVPI$. In our example, we obtain $E = .55/1.2 = .4583$. Loosely speaking, this means that the sample information is about 45% perfect. Note that the value of sample information depends on the strength of the link between the indicators and the state of nature. If the indicators are very strong, the efficiency will be close to one, if they are very weak, it will be close (or equal to) zero. Since *EVSI* is never negative and it can never exceed the value of *EVPI*, the efficiency is always a number between 0 and 1. Loosely speaking, it indicates the value of sample information as a proportion of perfection.

Consider now the same example with two different indicators, whose strengths are shown in Table 9.6.

Table 9.6: Conditional probabilities $P(I|s)$

	s_1	s_2	s_3
I_1	.9	.6	.2
I_2	.1	.4	.8

The indicator probabilities and posterior probabilities for this example are calculated and shown in Tables 9.7 and 9.8 for the indicators I_1 and I_2, respectively.

Table 9.7: Computation of $P(I_1)$ and $P(s|I_1)$

s	$P(s)$	$P(I_1\|s)$	$P(I_1\|s)\,P(s)$	$P(s\|I_1)$
s_1	.5	.9	.45	.6716
s_2	.3	.6	.18	.2687
s_3	.2	.2	.04	.0597
			$P(I_1) = .67$	

Table 9.8: Computation of $P(I_2)$ and $P(s|I_2)$

s	$P(s)$	$P(I_2\|s)$	$P(I_2\|s)\,P(s)$	$P(s\|I_2)$
s_1	.5	.1	.05	.1515
s_2	.3	.4	.12	.3636
s_3	.2	.8	.16	.4848
			$P(I_2) = .33$	

Constructing a decision tree similar to that in Figure 9.8 and performing the backward recursion results in $EPII = 2.12$, so that $EVSI = 2.12 - 1.5 = .62$ and $E = .62/1.2 = .5167$, slightly more efficient than in the original example.

In case the indicators are totally random, we would expect the value of this information to be zero. It can be readily seen that this is indeed the case. To illustrate this, consider again the above example and suppose now that there are three indicators. The conditional probabilities $P(I_k|s)$ all equal ⅓. This results in all indicator probabilities equaling ⅓ as well, while the posterior probabilities equal the prior probabilities. Inserting them into the decision tree, we find that in the first step of the backward recursion, we obtain the same expected payoffs we would as if Bayes's rule were used and, in the next step when making the decision, we will choose the best Bayesian decision with a payoff of EMV^* in each of the three cases. The next step will multiply this value by the indicator variables by ⅓, resulting again in EMV^*, so that $EPII = EMV^*$ and $EVSI = 0$.

An interesting case occurs in the other extreme. It is apparent that if we were to use a forecasting institute whose forecast is always correct the value of this information equals the value of perfect information. On the other hand, imagine an institute whose advice is always wrong. At first glance it would appear that the value of such advice equals zero. This is, however, not correct. As a matter of fact, such information is also perfect, as we can rely on it: whenever they say one thing, we know that the opposite applies. It is the consistency between what is predicted and what actually happens that really counts.

9.6 Utility Theory

Utilities have been used for a long time by economists. Particularly noteworthy are the analyses by the psychologists Kahnemann and Tversky in the 1970s. The main idea is to express the usefulness of a product or a service to the individual decision maker. In order to illustrate the concept, consider the following argument. If the expected value were to apply, then a decision maker would be indifferent to the choice of either a certain $50,000 gift and a lottery that pays $100,000 with a 50% chance and a zero payoff with a 50% chance. The expected value in both cases is the same: $50,000. However, most decision makers would prefer the certain $50,000.

Let us then take this argument a step further. Suppose we were to offer the aforementioned lottery—a $50,000 payoff with a 50% chance and a zero payoff with a 50% chance—to a decision maker and inquire what amount of money received with certainty he were to consider equivalent to playing the lottery. this value is called the *certainty equivalent*. The certainty equivalent is typically determined by a string of questions that narrow down the value. For instance, we would describe the lottery to the decision maker and offer, say, $45,000 for certain. Would he take the $45,000? If so, we renege on our offer and offer only $40,000 instead. This process continues until the certainty equivalent is found. For many people, the certainty equivalent is quite low, some go as low as $20,000. This shows a behavioral trait referred to as *risk aversion*. As a rule, if a decision maker's certainty equivalent is less than the expected value of the lottery, the decision maker is *risk averse*. If the certainty equivalent is higher than the expected value of the lottery, the decision maker is *risk seeking* (gamblers are a typical example), while if a decision maker's certainty equivalent equals the expected value of a lottery, he is called *risk neutral*. The graph in Figure 9.9 plots a dollar value against the decision maker's certainty equivalent of the lottery.

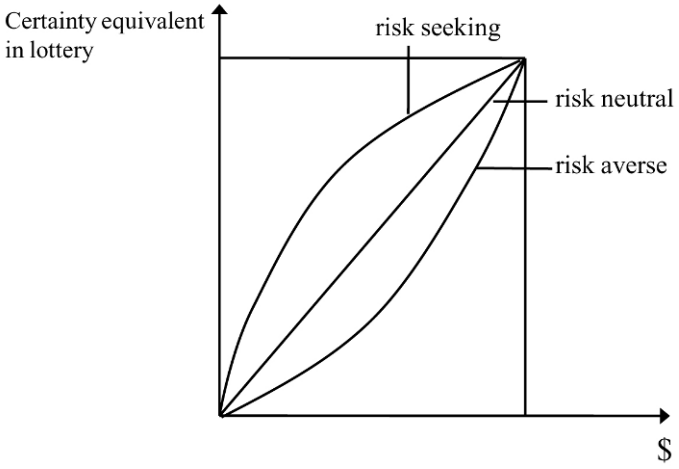


Figure 9.9

Note that it is important to realize when asking these questions, that many times people who claim they would rather take a risk and gamble than accept a fairly small certainty equivalent, these people would change their mind once the actual amount is put in front of them, taking the decision out of a purely "theoretical" realm into the hands-on practical world. Also note that the certainty equivalents are specific to a decision maker and are not transferable, as individuals differ in their acceptance of risk.

Once the certainty equivalents have been determined, they can be used to replace the actual payoffs, so that the problem is then to maximize the expected utility. This concept is also able to deal with cases, in which information concerning the likelihood of states of nature is known, but may be ignored by the decision maker. As an example, consider the case of a patient, whose physician has the choice of a number of drugs. Assume that one drug may be able to reduce the pain somewhat without known side effects, while another may not only eliminate the pain, but also the cause—with the possibility of major side effects that include death. If the latter event has only a tiny probability of occurring, the expected "payoff" to the patient may be such that the second and more effective drug may be chosen. However, the physician may choose to either ignore the probabilities and use Wald's rule so as to minimize the worst-case damage and choose the former less effective drug, or, similarly, may assign a very high negative "payoff" to the possibility of major side effects, resulting also in the former drug being chosen. Assigning very high costs (typically shown as $M >> 0$) to options is a technique that is also used in linear programming under the name penalty costs, whenever options or situations are to be avoided, while still using objectives that maximize the sum of benefits.

Exercises

Problem 1 (influence diagram): Consider the following situation. Jill lives presently in Missoula, Montana, where she has a fairly boring job. She has heard that there are many opportunities in Denver, Colorado, and she plans to go there, possibly resettle there, and buy a small house for herself. She plans to use her annual vacation to look things over in Denver.

Develop an influence diagram and a decision tree for the problem.

Solution: The list of potential events and the two graphs below indicate just some possible set of interdependencies of the decisions and events.

Decision	Random event	Consequence
Travel to Denver and look for a job	Economic situation Get job offer	Salary (or net worth)
Resettle in Denver		
Purchase a house in Denver		

The influence diagram for this list is shown in Figure 9.10.

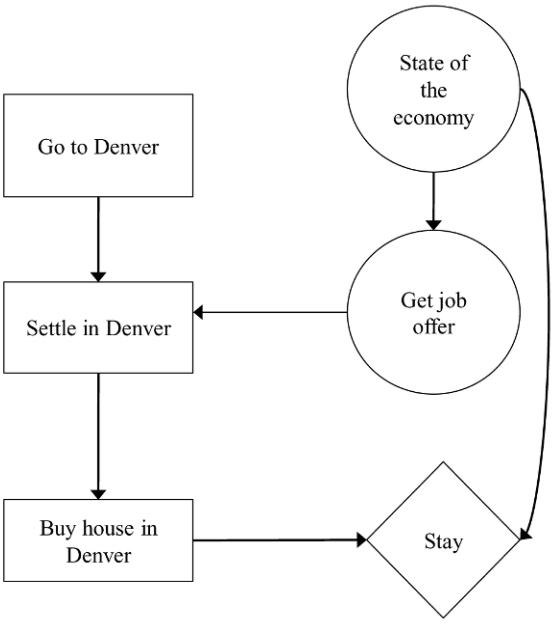


Figure 9.10

A possible decision tree that describes the sequence of events is shown in Figure 9.11.

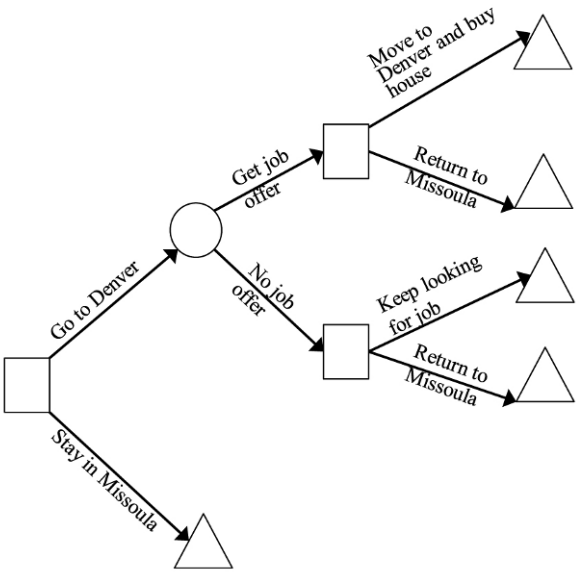


Figure 9.11

Problem 2 (single-stage decision-making under uncertainty and risk): Consider the payoff matrix of a game against nature.

	s_1	s_2	s_3	s_4	s_5	s_6
d_1	5	–2	7	1	0	–6
d_2	6	0	3	–5	8	1
d_3	1	4	0	0	–1	0
d_4	4	–2	3	–5	6	0

(a) Explain in one short sentence the concept of a dominated decision. Are there any dominated decisions in this example?
(b) How would an optimist decide?
(c) How would a pessimist decide?
(d) Find an optimal strategy using the minimax regret criterion.
(e) Suppose that the decision maker has been informed that the states of nature occur with probabilities of .3, .2, .1, .1, .05, .25. How would a risk-neutral decision maker decide?
(f) Construct the graph that plots the probabilities that a decision achieves a target value T for all target values between –10 and 10 for all decisions. What is the optimal decision for $T = 3½$? For $T = 5½$?

Solutions:
(a) A decision dominates another if it is better or the same for all states of nature. In this example, decision d_2 dominates d_4.

(b) An optimist anticipates payoffs of 7, 8, 4, and 6, so that d_2 would be chosen.
(c) A pessimist anticipates payoffs of 6, –5, –1, –5, so that d_3 would be chosen.
(d) The regret matrix is $\mathbf{R} = \begin{bmatrix} 1 & 6 & 0 & 0 & 8 & 7 \\ 0 & 4 & 4 & 6 & 0 & 0 \\ 5 & 0 & 7 & 1 & 9 & 1 \\ 2 & 6 & 4 & 6 & 2 & 1 \end{bmatrix}$ with anticipated maximal regrets of 8, 6, 9, 6, so that the (pessimistic) choice is d_2 or d_4 with regret of 6.
(e) The anticipated payoffs in Bayes's model are 0.4, 2.25, 1.05, and 0.9, so that the decision maker will choose d_2.
(f) d_1: solid line, d_2: broken line, d_3: dotted line, the dominated decision d_4 is deleted for clarity.

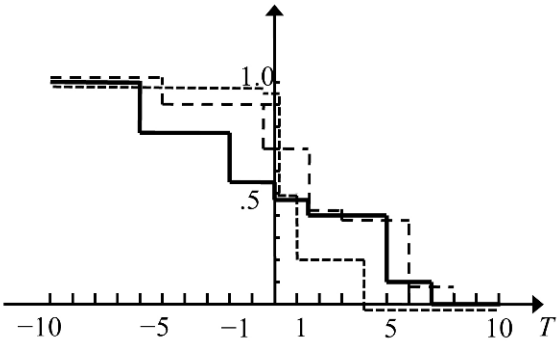


Figure 9.12

For $T = 3½$, decision d_1 is best, followed by d_2 and d_3. For $T = 5½$, d_2 is best, then d_1 and d_2.

Problem 3 (sensitivity analysis): Consider a decision problem with three decisions and four states of nature. The payoff matrix is as follows:

$$\begin{bmatrix} 2 & 0 & -2 & 4 \\ 1 & 1 & 2 & -3 \\ 2 & 3 & -2 & 1 \end{bmatrix}$$

Furthermore, suppose that the probabilities of the four states of nature are .4, .1, .3, and .2.

(a) Perform a sensitivity analysis on a_{34}, the payoff that results from decision d_3 coupled with the fourth state of nature. It has been estimated at $a_{34} \in [0, 4]$.

(b) Back to the original situation, perform a sensitivity analysis on p_2. It is assumed that (*i*) for each unit of increase of p_2, the probability p_1 decreases twice as much as each of p_3 and p_4, and (*ii*) that p_2 may decrease by as much as .05, while it may increase by at most .2.

Solution: (a) Given the payoff $a_{34} = 1 + \varepsilon$, we obtain expected monetary values of 1.0, 0.5, and $0.7 + 2\varepsilon$ for the three decisions. Given that $a_{34} \in [0, 4]$, or, equivalently, $\varepsilon \in [-1, 3]$, Figure 9.13 shows the expected monetary values of the three decisions as functions of ε and the upper envelope in form of the broken line.

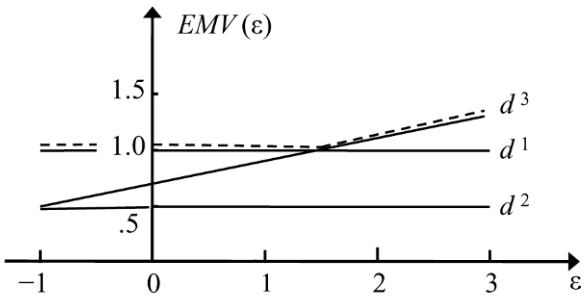


Figure 9.13

This leads to the following decision rule:
if $a_{34} \le 2.5$, (or, equivalently, $\varepsilon \le 1.5$), choose d_1,
if $a_{34} \ge 2.5$ (or, equivalently, $\varepsilon \ge 1.5$), choose d_3.

(b) The updated probabilities are $p = [.4 - \frac{1}{2}\varepsilon, .1 + \varepsilon, .3 - \frac{1}{4}\varepsilon, .2 - \frac{1}{4}\varepsilon]$, and the resulting expected monetary values are $EMV(\varepsilon) = 1.0 - 1.5\varepsilon$, $.5 + .75\varepsilon$, and $.7 + 2.25\varepsilon$ for the three decisions. Figure 9.14 shows the expected monetary values of the three decisions as functions of ε and the upper envelope is shown by the broken line.

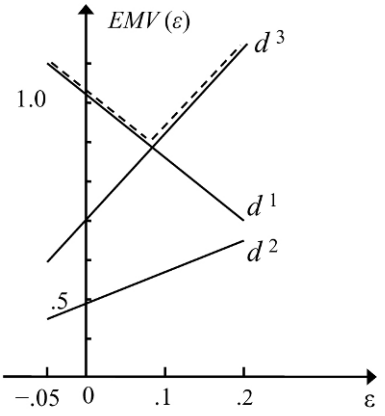


Figure 9.14

This results in the following decision rule:

if $p_2 \leq .18$ (or, equivalently, $\varepsilon \leq .08$), then choose d_1,
if $p_2 \geq .18$ (or, equivalently, $\varepsilon \geq .08$), then choose d_3.

Problem 4 (expected values of perfect and sample information): The Canadian McMoose chain of fast-food outlets is deciding how to keep up with the changing tastes of its customer base. They have narrowed down their choices to the following three decisions: d_1 = completely redecorate the existing franchises, d_2 = rebuild the outlets, and d_3 = modify the existing decor slightly to emphasize the "mooseyness" of the outlet. The chain faces different states of the economy s_1, s_2, and s_3: level, a slight upturn, a significant upturn. The payoffs for all combinations of decisions and states of the economy is shown in the following table:

	s_1	s_2	s_3
d_1	5	4	2
d_2	–4	2	9
d_3	3	8	1

The probabilities for the three states of the economy have been determined as .3, .2, and .5.

(a) Determine the expected payoffs for the three decisions and choose the most preferred decision on that basis.
(b) What is the expected value of perfect information?
(c) The McMoose management considers hiring a research institute to obtain more detailed information about the state of the economy. They use two indicators I_1 and I_2 for their forecast. These two indicators are linked to the state of the economy as shown in the following table of conditional probabilities $P(I|s)$:

	s_1	s_2	s_3
I_1	.5	.1	.8
I_2	.5	.9	.2

Construct the decision tree for this problem and determine the expected value of sample (imperfect) information. If the research institute charges 1.4 for their services, should they be hired? Explain in one very short sentence.

(d) What is the efficiency of the sample information?

Solution:

(a) The expected payoffs are 3.3, 3.7, and 3.0, so that they would choose d_2 and get 3.7.

(b) $EVPI = 7.6 - 3.7 = 3.9$.

(c)

For I_1:

s	$P(s)$	$P(I_1\|s)$	$P(I_1\|s)P(s)$	$P(s\|I_1)$
s_1	.3	.5	.15	.2632
s_2	.2	.1	.02	.0351
s_3	.5	.8	.40	.7018
			$P(I_1) = .57$	

For I_2:

s	$P(s)$	$P(I_2\|s)$	$P(I_2\|s)P(s)$	$P(s\|I_2)$
s_1	.3	.5	.15	.3488
s_2	.2	.9	.18	.4186
s_3	.5	.2	.10	.2326
			$P(I_2) = .43$	

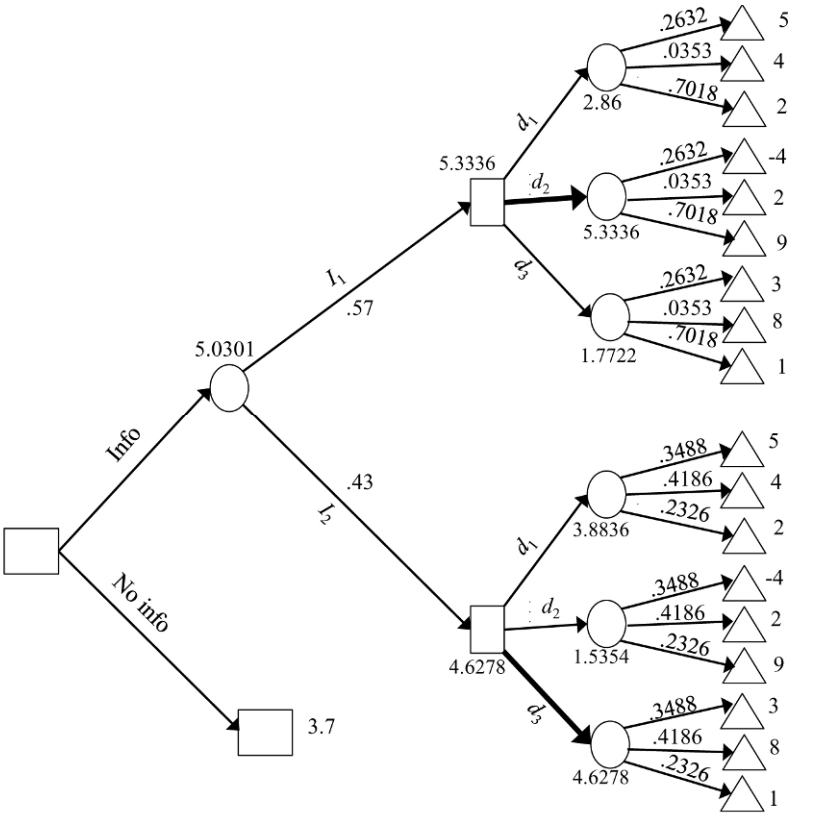


Figure 9.15

EVSI = 5.0301 – 3.7 = 1.3301, which is less than the amount requested by the institute, so do not hire them.

(d) The efficiency is $E = 1.3301/3.9 = .3411$.

Problem 5 (expected value of perfect and sample information): The Australian automobile manufacturer Australomobil must decide whether or not to manufacture the transmissions of their "Piticus" model in-house (d_1) or to subcontract them out (d_2). The company faces different levels of demand for the Piticus that are defined as s_1, s_2, s_3, and s_4. The payoffs for all combinations of decisions and levels of demand are shown in the following table:

	s_1	s_2	s_3	s_4
d_1	4	–1	–2	4
d_2	–4	2	1	3

(a) Are there any dominances in the payoff matrix? Explain in one short sentence.
(b) How would a pessimist decide? What is the optimal decision under the regret criterion? In both cases, what are the anticipated payoffs?
(c) Given prior probabilities of .2, .3, .4, and .1 for the states of nature, what is the optimal decision with Bayes's criterion? What is the expected payoff?
(d) What is the expected value of perfect information?
(e) The Australomobil management considers hiring a research institute to obtain more detailed information about the future level of demand. They use two indicators I_1 and I_2 for their forecast. These two indicators are linked to the level of demand as shown in the following table of conditional probabilities $P(I|s)$:

	s_1	s_2	s_3	s_4
I_1	.4	.8	.9	.5
I_2	.6	.2	.1	.5

Construct the decision tree for this problem & determine the expected value of sample (imperfect) information. If the research institute charges 0.7 for their services, should they be hired? Explain in one very short sentence.
(f) What is the efficiency of the sample information?

Solution:
(a) There are no dominances. For s_1, d_1 is better than d_2, but for s_2, d_2 is preferred over d_1.
(b) A pessimist will anticipate payoffs of –2 and –4, respectively. He will choose d_1 and anticipate a payoff of –2. For the regret criterion, we set up the regret matrix
$\mathbf{R} = \begin{bmatrix} 0 & 3 & 3 & 0 \\ 8 & 0 & 0 & 1 \end{bmatrix}$, so that the (anticipated) maximal regrets are 3 and 8. The decision maker will then choose d_1 and anticipate a regret of 3.
(c) The expected payoffs are .1 and .5, so that they would choose d_2 and get .5.
(d) $EVPI = 2.2 - .5 = 1.7$.
(e)

s	$P(s)$	$P(I_1\|s)$	$P(I_1\|s)P(s)$	$P(s\|I_1)$
s_1	.2	.4	.08	.1096
s_2	.3	.8	.24	.3288
s_3	.4	.9	.36	.4932
s_4	.1	.5	.05	.0685
			$P(I_1) =$ .73	

s	$P(s)$	$P(I_2\|s)$	$P(I_2\|s)P(s)$	$P(s\|I_2)$
s_1	.2	.6	.12	.4444
s_2	.3	.2	.06	.2222
s_3	.4	.1	.04	.1481
s_4	.1	.5	.05	.1852
			$P(I_2) =$ .27	

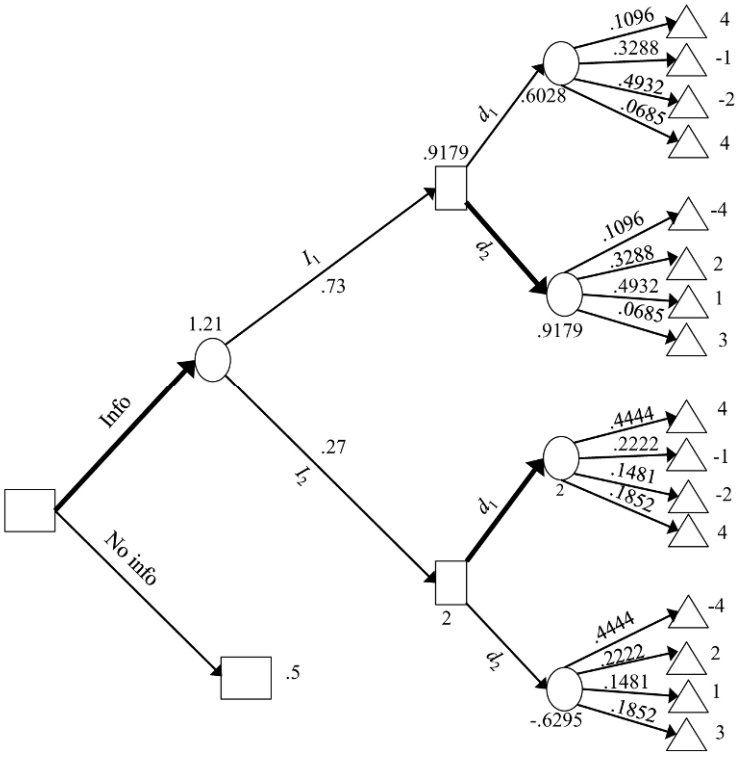


Figure 9.16

EVSI = 1.21 – .5 = .71, which is slightly more than the amount requested by the institute, so they should hire them.

(f) The efficiency is $E = .71/1.7 = .4176$.

10 Inventory Models

Worldwide, companies hold billions of dollars in inventories. The main reason is to create a buffer that balances the differences between the inflow and outflow of goods. Inventories can be thought of as water tanks: there may be a constant inflow of water that is pumped into the tank by a pump, while the outflow is low at night, high in the morning (when people get up, take a shower, etc), it then decreases significantly until the demand again increases in the evening (when people come home, do laundry, etc), just to fall off again for the night. Other, popular, examples include grocery stores whose inventories consist of various foodstuffs awaiting sale to its customers. Here, the delivery of the goods is in bulk whenever a delivery truck arrives, while the demand is unknown and erratic. In the case of hospitals, they have in stock medical supplies, bed linen and blood plasma. Again, the demand for these items is uncertain and may differ widely from one day to the next.

All these instances have a few basic features in common. They have a supply, a demand, and some costs to obtain, keep, and dispose of inventories. The next section will introduce a number of parameters and variables that are typically found in inventory models. Section 10.2 describes a basic inventory model, and the subsequent sections deal with a variety of extensions of the basic model.

10.1 Basic Concepts in Inventory Planning

For many organizations, inventories represent a major capital cost, in some cases the dominant cost, so that the management of this capital becomes of the utmost importance. When considering the inventories, we need to distinguish different classes of items that are kept in stock. In practice, it turns out that about 10% of the items that are kept in stock usually account for something in the order of 60% of the value of all inventories. Such items are therefore of prime concern to the company, and the stock of these items will need close attention. These most important items are usually referred to as "*A* items" in the *ABC* classification system developed by the General Electric Company in the 1950s. The items next in line are the *B* items, which are of intermediate importance. They typically

H.A. Eiselt and C.-L. Sandblom, *Operations Research: A Model-Based Approach*,
DOI 10.1007/978-3-642-10326-1_10, © Springer-Verlag Berlin Heidelberg 2010

represent 30% of the items, corresponding to about 30% of the total inventory value. Clearly, *B* items do require some attention, but obviously less than *A* items. Finally, the bottom 60% of the items are the *C* items. They usually represent maybe 10% of the monetary value of the total inventory. The control of *C* items in inventory planning is less crucial than that of the *A* and *B* items. The models in this chapter are mostly aimed at *A* items.

Due to the economic importance of the management of inventories, a considerable body of knowledge has developed as a specialty of operations research. We may mention *just-in-time* (*JIT*) systems that attempt to keep inventory levels in a production system at an absolute minimum, and put to work in Toyota's so-called *kanban* system. There is also *material requirements planning* (*MRP*) aimed at using the estimated demand for a final product in order to determine the need for materials and components that are part of a final product. *Multi-echelon* and *supply-chain management* systems also consider similar aspects of production-inventory control systems. Such topics are beyond the scope of this text, in which we can only cover some basic inventory models.

Throughout this chapter, we will deal with inventory models that concern just a single item. Consider an item for which the demand per period (typically a year, but other time frames can easily be accommodated) is known or estimated to be D units. Unless otherwise specified, the parameter D is assumed to be constant over time; if it is not, we will denote it by D_t with the subscript t indicating the time period.

The number of items in stock is depleted over time by the demand. On the other hand, the stock is also increased from time to time by additions caused by deliveries, referred to as *orders*. Typically, replenishments are assumed to be instantaneous (such as the arrival of goods by the truckload), resulting in sudden jumps in the inventory level, whereas deliveries to satisfy demand are typically assumed to be gradual. The order quantity is denoted by Q. In the models presented in this book, Q turns out to be constant over time, given that the parameters of the model do not change.

Related to the order quantity is also the *lead time* t_L. The lead time is defined as the time that elapses between the placement of an order and the moment that the shipment actually arrives and is available on the shelf.

It is useful to distinguish between three types of inventory/stock levels:

I^O denotes the *inventory on hand*. By this we mean stock that is physically on the shelf and immediately available to satisfy demand. Clearly, I^O must be nonnegative, i.e., $I^O \geq 0$.

I^N denotes the *net inventory* on hand, which is the inventory on hand minus *backorders*, the latter being unsatisfied amounts of demand due to insufficient

inventory on hand. The backordered demand will be instantly delivered whenever the stock is replenished and a sufficient number of items is on hand. The relation $I^N = I^O$ – (backorders) may be negative in case the backorders exceed the inventory on hand. If there are no backorders, then $I^N = I^O$.

I^P, denotes the *inventory position*, which is defined as $I^P = I^N$ + (outstanding orders).

Figure 10.1 may be used to visualize the differences between the different types of inventories. The inventory on hand I^O is shown by a solid line, the net inventory I^N by a broken line, and the inventory position I^P is shown by a dotted line.

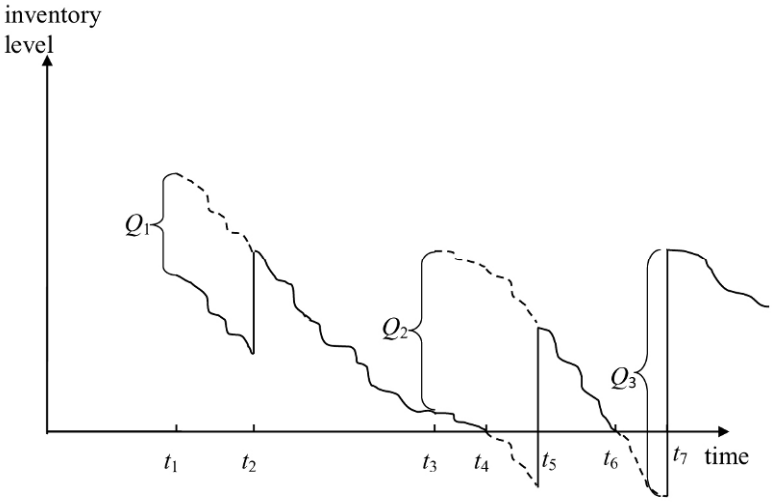


Figure 10.1

To the left of t_1, the inventory level I^O decreases until we place an order of magnitude Q_1 at time t_1. (The lead time is $t_2 - t_1$). At this point, I^O decreases further, while I^P jumps up by Q_1 and decreases parallel to I^O. At t_2, the shipment arrives, the inventory on hand jumps up and reunites with I^P. From this point on, the inventory on hand decreases until we place another order at t_3. The order is of magnitude Q_2, and the lead time is $t_5 - t_3$. The inventory position jumps up by Q_2 and decreases then parallel to the inventory on hand until t_4 occurs, when we run out of stock. From this point on, I^O equals zero, while I^N continues with negative inventory levels, while the inventory position is parallel to I^N. When the shipment arrives at t_5, the inventory level jumps up and all inventory curves are again united. From this point on, I^O decreases and reaches the zero level at t_6. The net inventory then continues in the negative inventory levels, while I^O remains at the zero level. At t_7, we place an order of magnitude Q_3 which we assume arrives

instantaneously, i.e., we have zero lead time. The inventory level then continues to rise and fall in similar fashion.

We then have the following I^O, I^N, and I^P values:

To the left of t_1: $I^O = I^N = I^P$,
between t_1 and t_2: $I^O = I^N$, $I^P = I^O + Q_1$,
between t_2 and t_3: $I^O = I^N = I^P$,
between t_3 and t_4: $I^O = I^N$, $I^P = I^O + Q_2$,
between t_4 and t_5: $I^O = 0$, $I^P = I^N + Q_2$,
between t_5 and t_6: $I^O = I^N = I^P$,
between t_6 and t_7: $I^O = 0$, $I^P = I^N$, and finally
to the right of t_7: $I^O = I^N = I^P$.

In the models we discuss below, the order size will usually be the same for all orders, and the lead time will also be the same for all orders, if not zero. The cost for carrying one unit of the good in inventory for one period is called the unit *carrying* or *holding cost*. We will denote this cost by the parameter c_h. Occasionally, the carrying cost c_h will be specified as a proportion of the inventory value rather than per unit of inventory. This will be clearly indicated whenever it applies. The total carrying cost is therefore related to the amount I^O of inventory actually on hand and will be zero for those time periods when $I^O = 0$. For some inventory models, backorders are not allowed, which implies that $I^N \geq 0$, whereas models with backorders (i.e., planned shortages) include the unit *shortage cost* c_s which is the cost charged to be out of stock by one unit for one period of time. The carrying costs c_h derive from capital cost, i.e., the cost of capital tied up in inventory, insurance, storage, security, heating/cooling, as well as costs for theft, damage, and obsolescence. In practice, the major component is the capital cost.

In contrast, the *ordering costs* c_o are defined as the cost of placing an order and having it delivered. It includes administrative costs as well as transportation costs. Ordering costs are usually considered to be independent of the size of the order, but since transportation costs are involved, one may question this assumption. However, we may argue that transportation costs may not be directly related to the size of the order: if the order is delivered by container or truck, the rate charged is usually not much dependent on whether the container is one-quarter or three-quarters full; a similar argument applied to truckloads. Additionally, if the transportation costs were to relate to each item, then—assuming that the demand must be satisfied—the total ordering cost for a period with a given demand D would be the same, regardless of whether there are many small or few large orders. Being a constant, this factor would then not affect the optimization.

10.2 The Economic Order Quantity (*EOQ*) Model

The most basic inventory model, the *economic order quantity* or *EOQ* model, has an interesting history. Although the *EOQ* formula was first published in 1913 by Ford Whitman Harris, it has been known under other names such as Camp's formula and Wilson's lot size formula in the 1920s and 1930s. An interesting historical account is provided in an article by Erlenkotter, indicating deliberate attempts in the past to deny Ford Whitman Harris the credit of originating the *EOQ*.

The assumptions of the basic economic order quantity are:

- The inventory consists of one single unperishable good held in one location,
- the demand rate for the item is constant over time and demand must be satisfied exactly,
- the item is ordered from a single supplier in the same amount each time. For now, we assume that replenishment is instantaneous, even though that is not really necessary as we will show later,
- there are no quantity discounts,
- stockouts are not allowed and the demand must be satisfied completely, and
- the planning horizon is infinite and all model parameters are stationary, i.e., they do not change over time.

It is apparent that the problem has two components: *how much* to order and *when*. The question of when to order is simply resolved in case of instantaneous replenishments: since the net inventory is not allowed to become negative, and since it does not make sense for a replenishment to occur while net inventory is positive, it must be optimal to place an order (and immediately receive the shipment), when the inventory level reaches zero. Instead of looking at the time when to reorder, we use the inventory level as a proxy for the time clock. The three inventory levels I^O, I^N, and I^P are all equal and can be denoted by I.

Since the order size was assumed to be the same each time, the inventory levels over time will have the characteristic sawtooth pattern shown in Figure 10.2, where the time between two consecutive replenishment times is referred to as the *inventory cycle length* t_c, while the *order quantity* is Q.

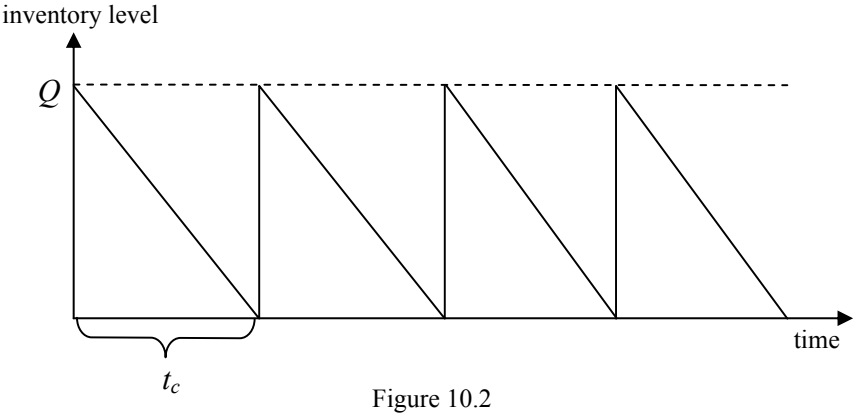


Figure 10.2

We now wish to minimize the total inventory-related costs *TC* per period (year), which is the sum of total ordering costs for the period and the total holding costs for the period. We could add the purchasing costs as well, but since we have assumed that the demand must be completely satisfied and there are no price discounts, the purchasing costs would be a term pD with p denoting the unit price of the good. This is a constant, and as such it will not affect the optimization and can thus be deleted in the process.

First, consider the ordering costs. If we were to place N orders, each of size Q, within the planning period, then the total amount ordered is NQ, which must equal the demand D: since stockouts are not allowed, NQ cannot be less than D, and with $NQ > D$ we would be carrying more inventory than needed, incurring unnecessary costs. With $NQ = D$, we find that $N = D/Q$ equals the number of orders per period. Incidentally, since t_c is the length of one inventory cycle, of which there are N in one period, $t_cN = 1$ period, so that $t_c = 1/N = Q/D$. Recalling that the cost for one order is c_o, we conclude that total annual ordering costs are $c_oN = c_oD/q$.

Since holding costs are charged per unit in inventory (the vertical axis in the diagram in Figure 10.2), we will need to compute the total area under the sawtooth curve, which can be seen to be $\frac{1}{2}Q$. In other words, the average inventory level is $\frac{1}{2}Q$, implying that the total annual holding costs are $c_h(\frac{1}{2}Q)$. Hence, the total inventory costs per period are

$$TC = c_oD/Q + \tfrac{1}{2}c_hQ.$$

This expression makes intuitive sense: the larger the order quantity Q, the lower the total ordering costs, since larger, but fewer, orders are being placed. However, the holding costs will be higher, since more inventory is being kept in stock (on average). This is illustrated in Figure 10.3, where the abscissa measures the order quantity, the ordinate measure the costs. The broken line (a rational function

of type "constant divided by the order quantity Q") represents the ordering costs, while the linear function (the broken line) represents the inventory holding costs. the solid line is then the sum of the two components.

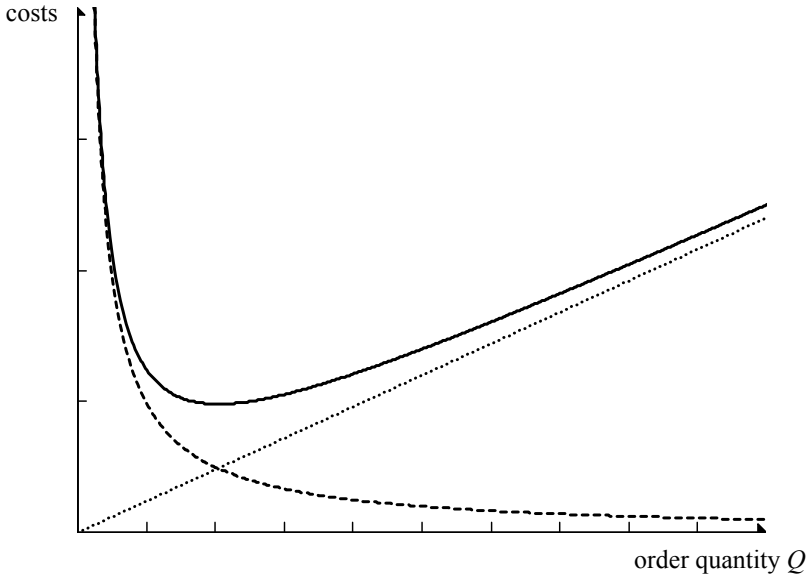


Figure 10.3

To determine the value Q^* that minimizes total inventory cost TC, we find the derivative with respect to Q, which results in

$$TC' = -c_o D/Q^2 + \tfrac{1}{2}c_h.$$

Setting it equal to zero results in the optimum order quantity

$$Q^* = \sqrt{\frac{2Dc_o}{c_h}}$$

(Technically, we also have to check the second derivative in order to ensure that the optimal order quantity results in minimal, rather than maximal, inventory costs. It does.) The expression for Q^* is referred to as the *economic order quantity EOQ*. Inserting this order quantity into the cost function, we obtain the inventory costs at optimum, which are

$$TC(Q^*) = \sqrt{\tfrac{1}{2}Dc_o c_h} + \sqrt{\tfrac{1}{2}Dc_o c_h} = \sqrt{2Dc_o c_h}\ .$$

Note that it just so happens that holding and ordering costs are always equal at optimum regardless of the value of the parameters D, c_o, and c_h; however, the solution Q^* is *not* optimal because holding and ordering costs are equal.

Given Q^*, we can then determine the related variables $N^* = D/Q^*$ and the optimal inventory cycle length $t_c^* = 1/N^* = Q^*/D$.

Example: A retail store faces a demand of D = 800 car battery chargers per year. The cost of placing an order for the chargers is c_o = \$100 and the holding cost is c_h = \$4 per charger per year. The economic order quantity is therefore $Q^* = \sqrt{\frac{2(800)(100)}{4}} = 200$ chargers and the total cost is $TC^* = \sqrt{2(800)(100)(4)} =$ \$800. The optimal number of orders to be placed throughout the year is $N^* = D/Q^* = 800/200 = 4$ orders, and the optimal inventory cycle length is $t_c^* = Q^*/D =$ 200/800 = ¼ year = 3 months. The total ordering costs are then 4(100) = \$400, which is half of the total cost, the other half is made up by the total holding costs.

A useful feature of the economic order quantity is its insensitivity to errors in the input data (i.e., the parameters). In our example, assume that the annual demand was erroneously estimated to be 920 chargers, instead of the correct amount of 800 chargers, i.e., a 15% overstatement. Using the *EOQ* formula we obtain the nonoptimal and erroneous value Q^* from the expression $Q^* = \sqrt{\frac{2(920)(100)}{4}} \cong 214.5$, which is an overstatement by slightly more than 7%, only about half of the original relative error. One can show that due to the square root of the formula, relative input data errors result in relative *EOQ* errors of only about half the size, for reasonable errors (say, 30% or less). Checking the total inventory costs that result from the wrong data, we find (using the true value of D = 800 and the erroneous order quantity Q^* = 214.5) that $TC(Q^* = 214.5) = \frac{100(800)}{214.5} + ½(4)(214.5) =$ \$801.96, which deviates only very marginally from the true value of \$800.

10.3 The Economic Order Quantity with Positive Lead Time

We will now carry our discussion further and extend the model to the situation where a positive lead time t_L elapses from the moment an order is placed and until the time the quantity ordered has arrived and been added to the inventory.

It is apparent that the decision *when* to order will in no way affect the decision *how much* to order. In other words, the optimum order quantity Q^* still applies. Instead of looking at the clock time for the point at which to place an order, we

will observe the inventory level and determine the order time in terms of the *reorder point R*: when the on-hand inventory level I^O decreases to the level R, an order is placed, which will arrive after a delay of t_L time units; at this point, the inventory level, for optimal performance, should have reached zero. We will consider two different cases, depending on the length of the lead time t_L, which differ on the basis of the relation between lead time and cycle time.

Case 1: $t_L \le t_c^*$, i.e., the lead time is less than or equal to the optimal inventory cycle length. The demand during the lead time is then $t_L D$, and it follows that if an order of size Q^* is placed when the inventory level reaches $R^* = t_L D$, then the replenishment will arrive exactly at the time when the inventory on hand has been depleted, which is neither too soon, nor too late. This situation is depicted in Figure 10.4.

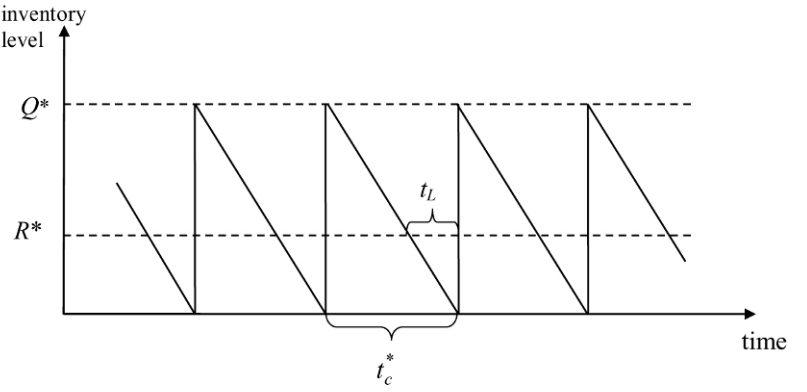


Figure 10.4

Case 2: $t_L > t_c^*$, i.e., the lead time is greater than the optimal inventory cycle length. The demand during the lead time is still $t_L D$, but since $t_L > t_c^*$, it follows that $t_L D > t_c^* D = Q^*$, which is the highest level of inventory on hand that we will ever reach. The arrival of an order will therefore occur during a subsequent inventory cycle and not during the cycle in which it was ordered. This situation is illustrated in Figure 10.5, where $t_c^* < t_L < 2t_c^*$.

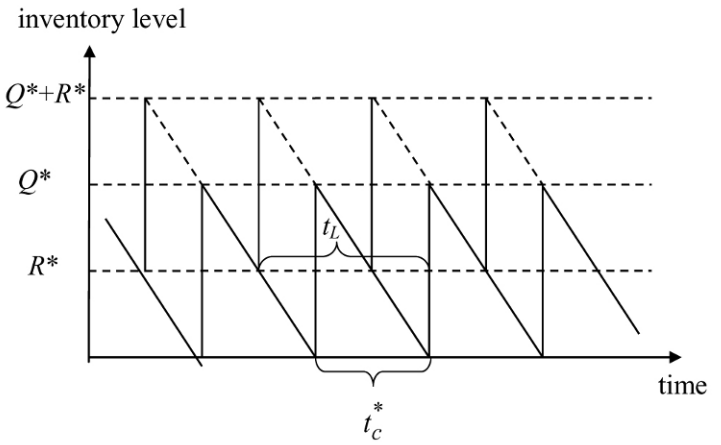


Figure 10.5

In general, there could be several replenishments occurring during the lead time. This number is actually $\left\lfloor \frac{t_L}{t_c^*} \right\rfloor$ (the "floor of the number"), i.e., the ratio $\frac{t_L}{t_c^*}$ rounded down to the nearest integer. In Figure 10.5, the floor equals 1, so that the replenishment arrives in the inventory cycle following immediately after the one during which it was ordered. In general, we obtain the relation

$$R^* = t_L D - \left\lfloor \frac{t_L}{t_c^*} \right\rfloor Q^* .$$

It can easily be demonstrated that this expression will cover both cases above. Since for $t_L < t_c^*$ (as in Case 1), $\left\lfloor \frac{t_L}{t_c^*} \right\rfloor = 0$, so that $R^* = t_L D$, which is the expression derived for that case.

Example: Consider the battery charger example above with $D = 800$, $c_o = \$100$, and $c_h = \$4$, for which we have obtained an optimal order quantity of $Q^* = 200$ at an annual cost of $TC^* = 800$. Given now a lead time of 2 months, i.e., 1/6 of a year, we have $t_L = 1/6 < ¼ = t_c^*$, so that Case 1 applies. Here, we find the optimal reorder point as

$R^* = t_L D = 1/6(800) = 133⅓$ units.

On the other hand, with a lead time of t_L = 4 months, i.e., ⅓ of a year, we have t_L = ⅓ > ¼ = t_c^*, so that Case 2 applies, and we find that

$$R^* = \tfrac{1}{3}(800) - \left\lfloor \frac{\frac{1}{3}}{\frac{1}{4}} \right\rfloor (200) = 66\tfrac{2}{3} \text{ units.}$$

As mentioned earlier, these shipments will not arrive in the inventory cycle they are ordered in, but in the next cycle.

A practical way to implement the reordering in Case 1 above where $t_L \le t_c^*$ is the *two-bin system*. Upon replenishment, the order quantity Q^* is physically separated and put into two storage bins. The first bin has a capacity of $Q^* - R^*$ units, while the second bin holds R^* units. The demand for items in stock is then satisfied exclusively from the first bin until it is empty. At that time, only the R^* items in the second bin remain and the inventory manager will reorder the item. From this point onwards, all subsequent demand is now satisfied from the second bin. Given the assumptions made in this section, the second bin will be depleted exactly at the time when the next shipment arrives and the process is repeated.

A similar argument allows the two-bin system to be implemented in Case 2 with $t_L > t_c^*$. Supermarkets use a "virtual" version of the two-bin system by electronically monitoring inventory levels by counting items that pass through the checkout counters. Orders are then automatically triggered when the appropriate reorder point has been reached.

10.4 The Economic Order Quantity with Backorders

Assume now that we allow backorders, so that the net inventory level I^N may become negative, in the sense that unsatisfied demand is recorded or "backordered," to be satisfied immediately upon replenishment of the inventory. Such planned shortages are considered to incur a shortage cost per unit and per period. These unit shortage costs will be denoted by c_s. These costs consist of the inconvenience to the customer of the unsatisfied demand and they will be difficult to estimate in practice. The costs could also include special handling costs that are incurred due to the preferential delivery to the customer of the backordered units when they become available. This will also imply that net inventory I^N and inventory position I^P are the same, i.e., $I^N = I^P$. Assuming a repetitive situation, the net inventory level will be as in Figure 10.6, where S denotes the amount of the maximal shortage. Note in the figure that the maximal inventory directly after replenishment is no longer Q as in the standard model, but $Q - S$, as the stockouts are satisfied first before new inventory is built up.

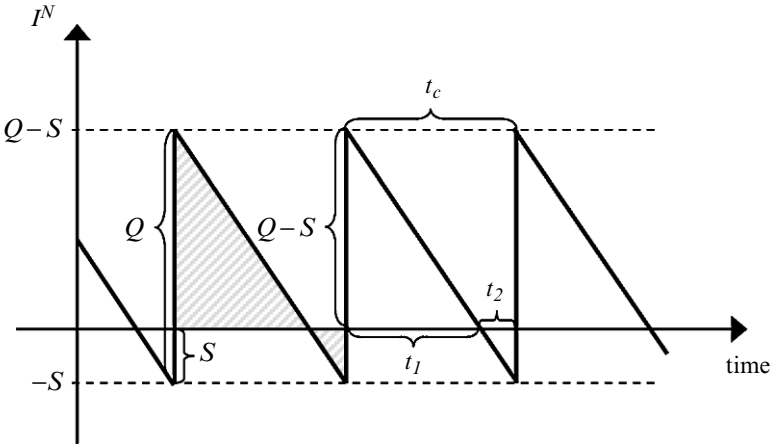


Figure10.6

In Figure 10.6, the parameter t_1 denotes the length of time during an inventory cycle during which the net inventory I^N is nonnegative, i.e., when there is no stockout. On the other hand, the parameter t_2 denotes the length of time during which there is no stock at hand. Clearly, $t_1 + t_2 = t_c$. Using the geometric relationship of the two similar (shaded) triangles in Figure 10.6, we find that $t_1/t_2 = (Q - S)/S$. We will use this expression below.

We will now determine optimal levels of the order size Q and the optimal largest shortage level S simultaneously by minimizing total inventory costs. These costs now include not only ordering and holding costs as before, but also shortage costs. As in the standard economic order quantity, the annual ordering costs are c_oD/Q. As far as the carrying costs are concerned, we find that the average inventory level is obtained by averaging the inventory level during the time that no stockouts occur. This weighted average is $\frac{1}{2}(Q-S)$ during the time t_1, while the inventory level during the time stockouts occur is zero for the duration t_2. After some calculations, this leads to inventory holding costs of $c_h \dfrac{(Q-S)^2}{2Q}$.

We can now deal with the average shortage in a similar fashion. The average annual shortage is $\frac{1}{2}S$ during the time t_2, which is when we have shortages. This leads to total shortage costs of $c_s \dfrac{S^2}{2Q}$. The total inventory costs are therefore

$$TC(Q, S) = c_o \frac{D}{Q} + c_h \frac{(Q-S)^2}{2Q} + c_s \frac{S^2}{2Q}.$$

Using partial derivatives, we can show that the total inventory costs are minimized for

$$Q^* = \sqrt{\frac{2Dc_o}{c_h}\frac{c_h + c_s}{c_s}} \text{ and } S^* = \sqrt{\frac{2Dc_o}{c_s}\frac{c_h}{c_h + c_s}} = \frac{c_h}{c_h + c_s}Q^*.$$

Example: Using the basic *EOQ* example above with its parameters $D = 800$, $c_o = \$100$, and $c_h = \$4$ to which we add unit shortage costs of $c_s = \$6$ per unit and year, we can then determine the optimal order quantity as $Q^* = \sqrt{\frac{2(800)(100)}{4}\frac{4+6}{6}} \cong 258.20$ units. The optimal shortage can then be determined as $S^* = \frac{c_h}{c_h + c_s}Q^* \cong 103.28$. The total costs are then $TC(Q^*, S^*) = 309.84 + 185.90 + 123.94 = \$619.68$.

Note that while this model includes more cost items than the basic *EOQ* model, the same demand and cost parameters cost less in this model. The reason is that this model allows the decision maker an added possibility, *viz.* to run shortages. Nobody says that we have to have shortages (this model certainly allows not having any), but here it is cost-effective to plan some shortages. As a matter of fact, the relation between the unit shortage cost and the unit holding cost will determine the magnitude of the planned shortage: if c_s/c_h is large, then shortages are expensive and the solution will include only minor shortages. If, on the other hand, c_s/c_h is small, then shortages are comparably cheap, and the model will prescribe large shortages.

To push that argument even further, suppose that shortage costs would increase beyond all reasonable limits, i.e., $c_s \to \infty$. Then the "correction factor" $\frac{c_h + c_s}{c_s}$ in the order quantity root will approach 1, so that the order quantity Q^* will assume the same value as in the basic *EOQ* model. At the same time, the magnitude of the planned shortage will tend to zero, as the shortage cost appear only in the denominator of the formula. Thus it becomes clear that the basic model is just a special case of the model with shortages, given that shortage costs are infinitely high. This result is nothing but an application of the usual principle of penalty costs: if there is something that we do not want, assign a very high penalty to it, and as a result, the optimizer will not include the very expensive option in the solution.

10.5 The Economic Order Quantity with Quantity Discounts

So far we have assumed that the unit purchasing cost p is constant and independent of the order size Q. Recall that in the original economic order quantity model, the costs actually were the sum of ordering, holding, and purchasing costs, *viz*.,

$$TC(Q, p) = c_o D/Q + \tfrac{1}{2} c_h Q + pD.$$

However, we argued, for a fixed price p, the purchasing costs during the period under consideration are pD, which is a constant, which does not influence the solution, so that we could (and did) ignore the purchasing costs. In practice, however, many suppliers will offer incentives for purchases of larger quantities in the form of lower unit costs. The basic economic order quantity model described in Section 10.2 can easily be modified to take such quantity discounts into consideration. For simplicity, we will restrict ourselves to the standard model with no shortages allowed. To simplify our discussion, we assume that there are three price levels, the original non-discounted price and two discount levels. It is straightforward to extend the model to any number of discount levels.

Before we proceed, though, we have to make a minor modification. Recall that we said earlier that the main component of the holding costs are the cost for tied-up capital. Given that we paid a fixed price for the good so far, this cost could simply be expressed as a dollar amount for each unit in stock. Given that we now are paying a price that is no longer fixed but does depend on the discount level that we choose, the unit holding costs have to be redefined. This is most easily done by defining c_h as a proportion of the unit purchasing price p. Given that, the economic order quantity is redefined as

$$Q^* = \sqrt{\frac{2Dc_o}{c_h p}}\,.$$

We will denote the given (non-discounted) price level as p_0, the price with the small discount as p_1, and the price given the large discount as p_2. Clearly, $p_0 > p_1 > p_2$. The rationale behind this scenario is simple. For the regular price of p_0, we can obtain any quantity we desire. In order to convince our supplier to sell the goods for us at the lower price of p_1, we have to purchase at least a certain quantity of goods. This quantity will be called Q_1. Going one step further, we can ask our supplier to let us have the goods even cheaper at a price of p_2, which he may agree to, but only if we order at least Q_2 units with the obvious condition that $Q_2 > Q_1$.

At this point, we have a cost function for each of the price levels. This situation is shown in Figure 10.7.

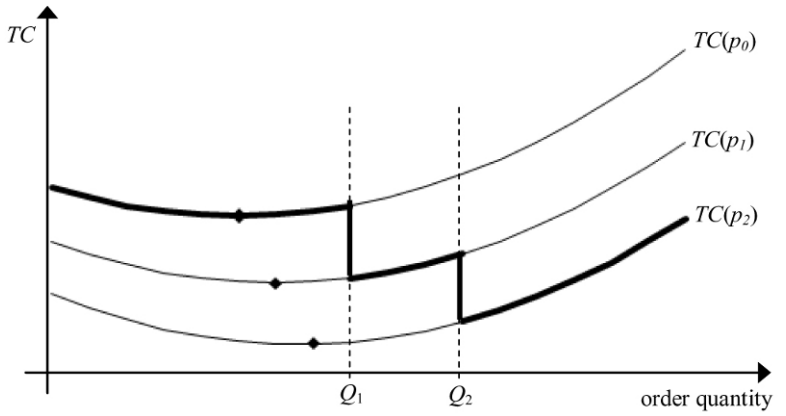


Figure 10.7

The three cost functions are shown as $TC(p_0)$, $TC(p_1)$, and $TC(p_2)$, respectively. The dots on the cost functions denote their respective optimal points, at which the costs are minimized. The minimal quantity levels that allow for the discounts are also shown.

Consider now the actual costs that we incur as the order quantity gradually increases. For very low order quantities, we must be on the highest cost curve, as we do not qualify for a discount. As we increase the value of Q, the costs decrease and reach their minimum at the dot on the $TC(p_0)$ curve. As Q increases further, the costs increase as well until we reach Q_1. At this point, we do qualify for the small discount, so that our actual costs jump down onto the cost curve $TC(p_1)$. As this point happens to be to the right of the minimum on this function, the costs increase as Q increases. This process continues until we reach Q_2, the value that allows us to obtain the second (larger) discount. Again, the costs drop at this point onto the third and lowest cost function $TC(p_2)$. Increasing the value of Q further, increases the total costs.

The piecewise nonlinear cost function is shown as a bold line in Figure 10.7. In order to determine the order quantity with the overall minimal costs, we have to examine each cost curve separately. More specifically, we determine the optimal order quantity at each price level, and then compare them and choose the option with the lowest costs.

First consider the highest cost curve without a price discount. We simply determine the point of lowest cost with the *EOQ* and record the associated cost. This is the optimal solution given the option of paying the regular price p_0.

We then continue to examine the costs incurred when paying the price p_1. Again, we determine the optimal quantity at this price by solving the economic order quantity with this price. (Note that with decreasing prices, the optimal order quantity increases slightly, as the expression in the denominator $c_h p$ increases). We then have to determine whether or not this quantity permits us to obtain the discount. If so, we have found our optimal order quantity at this level. If this is not the case, we have to move out of the optimum, but, as the function is increasing the farther we move out of the optimum, just as much as required to qualify for the discount.

In our illustration in Figure 10.7, the optimal order quantity is less than Q_1, the lowest quantity that qualifies for the price p_1. We thus increase the order quantity to Q_1 and determine the costs at that point. This is the optimal order quantity *given the price* p_1.

This process is repeated for all discount levels. Once this has been accomplished, we simply compare the best known costs at each price level and choose the overall minimum. This is our optimal solution.

This process can be illustrated by the following numerical

Example: A company faces an annual demand for 10,000 footballs. The purchasing costs are \$2 per football, the holding cost are 5% of the purchasing price per football and year, while the costs of placing one order are \$80. The supplier now offers a ½% discount in case the company orders at least 6,000 units. As an alternative, the supplier also offers a 1% discount, if the company orders at least 15,000 units. Consider all alternatives, compute the total costs in each case and make a recommendation.

The parameters of the problem include D = 10,000, c_h =5% of p, and c_o = \$80.

Case 1: No discount, so that p_0 = \$2. Then c_h = \$0.10, and we use the *EOQ* to compute the order quantity as Q^* = 4,000 with costs of TC^* = 200 + 200 + 20,000 = \$20,400.

Case 2: Small discount, so that p_1 = \$1.99. Then c_h = \$0.0995, and the solution of the *EOQ* is Q^* = 4,010.038. This quantity does not qualify for the discount, so that we have to move out of the optimum just as much as necessary to qualify for the discount. Hence we set Q := 6,000, for which we then obtain costs of TC(6,000) = 133.33 + 298.50 + 19,900 = \$20,331.83.

Case 3: Large discount, so that p_2 = \$1.98. Then c_h = \$0.099, and the solution of the *EOQ* is Q^* = 4,020.15. This quantity does not qualify for the discount, so that we have to move out of the optimum just as much as necessary to qualify for the discount. Hence we set Q := 15,000, for which we then obtain costs of TC(15,000) = 53.33 + 742.50 + 19,800 = \$20,595.83.

Comparing the three options, Case 2 offers the lowest total costs, so that we should order 6,000 footballs, obtain a ½% discount, and incur total costs of $20,331.83.

10.6 The Production Lot Size Model

As an alternative to ordering models of the type discussed so far in this chapter, we may produce the desired items ourselves. In such a case, the items will not arrive in one bulk as they do in case of orders, but they arrive one piece at a time from our machines. Assume for the time being that we cannot regulate the speed with which our machines produce the units: we either turn the machine on, in which they churn out r units per day (this is our production rate), or we turn off the machine, in which case we make nothing. Recall that our annual demand was assumed to be of magnitude D, from which we can easily compute the daily demand d, which is $D/365$, $D/360$, or $D/250$ (working days), depending on the decision maker's specifications. Before engaging in any computations, it is necessary to determine whether or not the system has any feasible solutions. The simple *regularity condition* is that $r \geq d$. If this condition is not satisfied, then it will not be possible to satisfy the total demand, and we have to find ways that allows us to do so. The following arguments assume that the regularity condition is satisfied.

Batch or *intermittent production* as described above occurs in many vertically integrated companies, where the ordered items are produced internally. A production run can then be considered an order, with the production run size corresponding to the order size Q, and the production setup cost corresponding to the ordering costs c_o.

Using an argument similar to that in Section 10.2, we note that D/Q is the number of setups or production runs per period, so that the total setup costs are $c_o(D/Q)$. As far as the carrying costs are concerned, we will consider the *production phase* t_r (the phase during with production and demand occur) and the *demand phase* t_d (the phase during which production does not occur, while demand occurs as usual) separately. In the production phase, inventory accumulates at the rate of $(r-d)$. We notice that the duration of the production phase is $t_r = Q/r$, so that the maximal level of inventory at the end of each production run will by $(r-d)Q/r$. During the demand phase, the inventory, that starts with a level of $(r-d)Q/r$, decreases to zero in linear fashion at a rate of d, so that the slope of the function in Figure 10.8 during that phase is $-d$). The average inventory level during the entire cycle of duration $t_c = t_r + t_d$ is then ½ $(r-d)q/r$. Therefore, the total carrying cost per period are $\frac{1}{2}c_h(r-d)Q/r$. As a result, the total production- and inventory-related costs are

$$TC = c_o D/Q + \tfrac{1}{2}c_h(r-d)Q/r$$

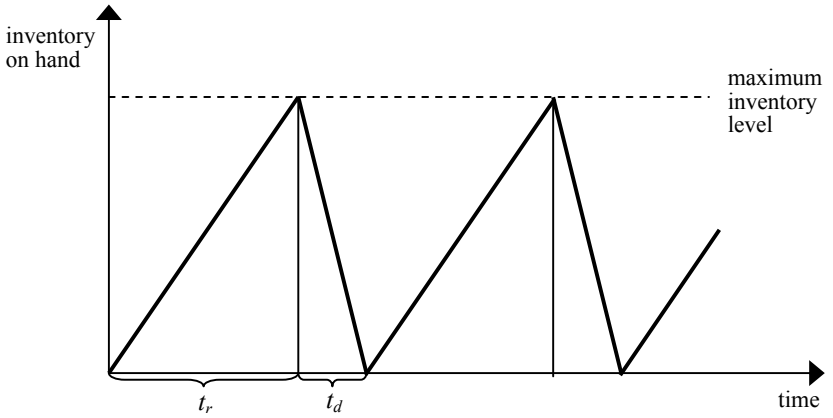


Figure 10.8

Following the same procedure applied to the standard economic order quantity in Section 10.2, we find the derivative with respect to the single variable Q, which results in $TC' = -c_o D/Q^2 + \frac{1}{2}c_h(r-d)/r$. Setting the derivative equal to zero results in the unique optimal lot size of

$$Q^* = \sqrt{\frac{2Dc_o}{c_h}\,\frac{r}{r-d}}$$

(Again, we have to check the second derivative so as to ensure that the lot size actually minimizes the costs. It does).

We can now illustrate the economic lot size by means of a numerical

Example: A bottling plant faces an annual demand of 200,000 bottles of a certain type. It can produce these bottles at a rate of 1,000 bottles per day during each of the 300 working days in a year. Setup costs for a production run are \$1,000, and each bottle has a carrying cost of 10¢ per bottle and year.

We first check whether or not the regularity conditions holds. Here, we have $r = 1{,}000 > 666.67 = 200{,}000/300 = d$, so that the condition is indeed satisfied. Thus we can compute the optimal quantity made during one production run as $Q^* = \sqrt{\frac{2(200{,}000)(1{,}000)}{0.10}\,\frac{1{,}000}{1{,}000-666.67}} \cong 109{,}545$ bottles. The corresponding costs are $TC(Q^*) = \$3{,}651.50$.

An interesting observation concerns the relation between the optimal lot size developed above and the economic order quantity. Given that we can equate the production setup cost and the unit order cost in the two respective models, we find

that the optimal value of Q in the lot size model is never smaller than the order quantity in the economic order quantity, and that the costs in the lot sizing model are never larger than those of the *EOQ*. As a matter of fact, the economic order quantity can be seen as a special case of the lot sizing model with an infinite production capacity (as exemplified by the fact that an order in the *EOQ* arrives with infinite speed). Increasing r to arbitrarily high values has the expression $r/(r-d)$ approach one, so that the lot size formula reduces to the standard economic order quantity. Applying this argument to our numerical example, we find that the economic order quantity with the same parameters as those used in the example equals $Q^* = 63{,}245.55$, a policy that costs $TC(Q^*) = \$6{,}324.56$.

Along similar lines, it is also interesting to note that more capable machines, i.e., those with higher production rates, incur higher costs. As a matter of fact, the machine with the "optimal" production rate has $r = d$, so that inventories are unnecessary, as customers satisfy their demand at the same rate the machine produces the goods. This is yet another example for optimal solution that fitting the production to the demand results in solutions with the lowest cost (for another example, see the "technology choice" example in the linear programming formulations).

In conclusion, just like the economic order quantity, the production lot sizing model has the attractive property of being robust, i.e., quite insensitive to changes of the parameters (input data).

10.7 The Economic Order Quantity with Stochastic Lead Time Demand

So far in this chapter we have assumed a deterministic environment, in which all relevant data are known with certainty, and in which the consequences of our actions are completely predictable. We will now extend our analysis to situations involving uncertainty and begin with the simple but important case, in which the demand during the lead time is a random variable. However, we assume that the demand during the lead time follows a discrete probability distribution that is known to us. The random behavior of the demand may cause undesired and unplanned stockouts and surpluses, which is shown in Figure 10.9. From the figure it is apparent that while the demand is irregular throughout, we are only concerned about the irregularity that occurs between the time that we have placed an order (i.e., after the reorder point has been reached), and the time that the next shipment arrives.

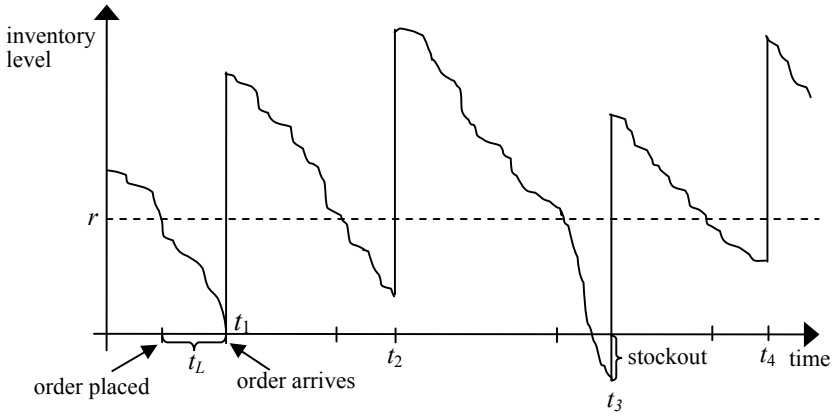


Figure 10.9

In Figure 10.9, inventory replenishment occurs at the time points t_1, t_2, t_3, and t_4, points in time that are not necessarily evenly spaced, as a consequence of the reorder point R having been reached at the points in time $t_1 - t_L$, $t_2 - t_L$, $t_3 - t_L$, and $t_4 - t_L$. At t_1, there is neither a stockout nor a surplus (a rather unlikely event); at t_2 and t_4 we have a surplus inventory when the next shipment arrives, while at t_3, there is an unplanned shortage. Whenever a surplus occurs, we have in fact carried more inventory than was actually needed, thus incurring unnecessary carrying costs, while in case of a stockout, there will be a penalty cost c_p charged for each unit we are out of stock. Note that c_p is assumed to be independent of the length of time that we are out of stock. This is in contrast to the shortage costs c_s for the backorder model of Section 10.4, where the shortage cost was defined per quantity unit *and* per time unit. The difference between c_p and c_s in the backorder model and this model is that the shortages in Section 10.4 were planned deliberately, while the shortages in this section occur because there is a higher-than-expected demand during the lead time t_L.

Formally, we define c_p as the penalty cost per unit and stockout. Furthermore, we have $\overline{D} = E(D)$ as the expected valued of demand per year, the lead time demand d_L (a random variable), and the expected value of d_L is $\overline{d}_L = E(d_L)$. The (discrete) probability distribution of the lead time demand d_L is $p(d_L)$, while $F(d_L)$ is the cumulative probability distribution of d_L. We will restrict our discussion to the case, in which d_L is a discrete random variable. Furthermore, in this simple model, the length of the lead time t_L is still deterministic, i.e., fixed and known to the decision maker. We also assume that $R - \overline{d}_L \geq 0$, i.e., on average, there is still a positive inventory level when replenishment occurs. If this condition were not to be required, we would, on average, run out of stock at the end of each cycle.

Therefore, we may regard the quantity $R - \bar{d}_L$ as the amount of stock that is kept at all times. For this reason, this quantity is usually referred to as the expected *safety stock* or *buffer stock*.

10.7.1 A Model that Optimizes the Reorder Point

The objective in this section is to minimize the sum of the carrying costs for the expected safety stock plus the expected penalty costs for stockouts. This sum will be denoted by $TC_1(R, Q)$, since it depends on the reorder point R as well as on the order quantity Q. To start, we will simply use the order quantity Q_{EOQ}, which was obtained independently of the reorder point by way of the economic order quantity. This can be justified because of the robustness of the economic order quantity formula. We then obtain the partial cost function

$$TC_1(R, Q_{EOQ}) = c_h(R - \bar{d}_L) + c_p\left(\frac{\bar{D}}{Q_{EOQ}}\right)\sum_{d_L > R}(d_L - R)p(d_L)$$

where $Q_{EOQ} = \sqrt{\frac{2\bar{D}c_o}{c_h}}$, and where the first part of the relation is the cost for carrying the safety stock. The summation in the second part of relation is taken over all instances, in which shortages occur, so that we compute the expected shortage level. Differentiating $TC_1(R, Q_{EOQ})$ with respect to R and setting the resulting expression to zero yields the condition for the optimal reorder point R^*, which is

$$P[d_L > R^*] = \frac{c_h Q_{EOQ}}{c_p \bar{D}}.$$

As $P[d_L > R^*] = 1 - P[d_L \le R^*] = 1 - F(R^*)$, we obtain

$$F(R^*) = 1 - \frac{c_h Q_{EOQ}}{c_p \bar{D}}.$$

Since we have assumed that d_L is a discrete random variable, its cumulative distribution function F will be a step function that assumes only discrete values in the interval [0, 1]. Therefore, it is unlikely that the right-hand side of the above equation will equal one of these discrete values. As a way out of this dilemma, we let R^* denote the smallest value that satisfies the inequality

$$F(R^*) \ge 1 - \frac{c_h Q_{EOQ}}{c_p \bar{D}}.$$

Note that we only have to consider the possible values of d_L for R^*.

In order to illustrate the above discussion, consider the following numerical

Example: Consider again the battery charger example of Section 10.2 with a demand of $D = 800$, ordering costs of $c_o = \$100$, and holding costs of $c_h = \$4$ per charger per year. Furthermore, assume that the penalty costs are $c_p = \$5$ per charger and stockout. Suppose that the expected annual demand is $\overline{D} = 800$. Suppose that the demand during lead time has the following probability distribution.

d_L (units)	$p(d_L)$	$F(d_L)$
70	.1	.1
75	.2	.3
80	.2	.5
85	.3	.8
90	.2	1.0

The economic order quantity in this example equals $Q_{EOQ} = 200$ units, so that $1 - \frac{c_h Q_{EOQ}}{c_p \overline{D}} = 1 - \frac{4(200)}{5(800)} = 0.8$, and since the smallest value of d_L with $F(d_L) \geq 0.8$ equals 85, we have $R^* = 85$. As the expected demand $\overline{d}_L = E(d_L) = \sum_x x p_L(x) =$ 81.5, the expected safety stock will equal $R^* - \overline{d}_L = 85 - 81.5 = 3.5$ units. The carrying cost for the expected safety stock is then $c_h (R^* - \overline{d}_L) = 4(85 - 81.5) =$ \$14, and the expected penalty cost is $c_p \left(\frac{\overline{D}}{Q_{EOQ}} \right) \sum_{d_L > R^*} (d_L - R^*) p(d_L) =$ $5 \frac{800}{200}(90 - 85)(0.2) = \$20$. Note that stockouts occur only if $d_L > R^* = 85$, which happens only in case $d_L = 90$, an occurrence that has a probability of 0.2.

10.7.2 A Stochastic Model with Simultaneous Computation of Order Quantity and Reorder Point

We can now refine the above model and determine the order quantity Q and the reorder point R simultaneously. For that purpose, we consider the expected total cost of ordering, carrying, and penalty, i.e,

$$TC_2(Q, R) = c_o \frac{\overline{D}}{Q} + c_h(½Q + R - \overline{d}_L) + c_p \left(\frac{\overline{D}}{Q} \right) \sum_{d_L > R} (d_L - R) p(d_L)$$

Using partial differentiation with respect to Q and to R and setting the result to zero yields

$$Q^* = \sqrt{\frac{2\overline{D}}{c_h}\left(c_o + c_p \sum_{d_L > R^*} (d_L - R^*) p(d_L)\right)} \text{ and}$$

$$F(R^*) \geq 1 - \frac{c_h \overline{Q}}{c_p \overline{D}}.$$

Again, it is understood that R^* is taken to be the smallest value that satisfies the inequality. The above two relations should be solved simultaneously, which is difficult, since Q^* and R^* appear in both. Instead, we will use an iterative procedure that shuttles between these two relations. It commences with an order quantity Q^*, uses the second of the two relations to determine a reorder point R^*, it uses this reorder point in the first relation to compute a revised value of Q^*, and so forth until the process converges and the numbers do not change anymore. As an aside, note again that if the penalty costs get very large, the order quantity reduces again to the standard economic order quantity of the basic model.

This process will be illustrated in the following numerical

Example: Consider again the situation of the example in the previous section with a demand of D = 800, ordering costs of c_o = \$100, holding costs of c_h = \$4 per charger per year, penalty costs c_p = \$5 per charger and stockout, and the above probability distribution of the demand.

Again, we obtain Q^* = 200, so that R^* = 85, just as in the previous procedure. Using the modified economic order quantity, we then find a revised value of Q^* as

$$Q^* = \sqrt{\frac{2(800)}{4}[100 + 5(5)(0.2)]} \approx 204.94 \text{ units.}$$

Using this revised order quantity in the latter of the two relations, we find that

$F(R^*) \geq 1 - \frac{(4)(204.94)}{5(800)} \approx .795$, so that R^* = 85 again, and thus the procedure terminates.

Comparing the results for Q^* and R^* of the simple model in the previous subsection and the refined approach in this subsection, we notice that in both cases the reorder point is R^* = 85 units, whereas the order quantity is Q^* = 200 units in

the simple model (obtained by using the standard economic order quantity), while it is not very different at 204.95 ≈ 205 units in the refined model. Again, this demonstrates the robustness of the economic order quantity formula.

10.8 Extensions of the Basic Inventory Models

This section will offer an outlook on some inventory policies. Following the standard terminology (which is in some conflict of the symbols that we have used so far), we define s as the reorder level (what we have referred to so far as the reorder point, i.e., the inventory level at which an order is placed), R as the intervals at which the inventory level is checked, and S as the inventory level we have directly after a replenishment.

We now distinguish between periodic and continuous review systems. In a *periodic review system*, we check the inventory levels at regular intervals R (e.g., hourly, daily, or weekly), while in a *continuous review system*, we continuously watch the inventory level.

An *order-point*, *order-quantity*, or (s, Q) *policy* involves continuous review (i.e., $R = 0$) at which time an order of a given magnitude Q is placed whenever the inventory reaches a prespecified reorder level s. An example of an (s, Q) policy is the two-bin system described in Section 10.3.

An *order-point*, *order-up-to-level*, or (s, S) *policy* is another continuous review policy. Its key is an inventory level S that is specified by the inventory manager. This is an inventory level to be attained directly after a shipment is received. So, once the reorder point s is reached, an order of size $S - s$ is placed, which then increases the inventory position level I^p to S. This may be a reasonable policy in case the demand is irregular, so that at the time that an order is placed the inventory level may suddenly dip below s, at which time the regular order quantity may then not be sufficiently large.

A *periodic review*, *order-up-to-level*, *replenishment cycle policy*, or (R, S) *policy* is a periodic review policy. At each review instant (which occurs at intervals of length R time units) an order is placed of a size that raises the inventory position level I^p to S.

In addition, there are hybrid policies such as the (R, s, S) policy, where at review time nothing is done, if the inventory level is above s, whereas if it is at or below the level s, an order is placed to increase the inventory level to the magnitude S.

Each of the above policies has its own advantages and drawbacks, and it depends on the practical situation at hand which one is the most appropriate choice. Typically, continuous review policies such as the (s, Q) and (s, S) policies are suitable for the A items in the ABC classification introduced at the beginning of

this chapter. For *B* and *C* items, the cost of continuous review of the inventory level may not be justified, so that periodic review policies may make more sense. For *C* items, the review interval length *R* may be set, so that the review is done less frequently for these items of minor value. Modern computerized inventory control systems are of great help with any inventory system.

Exercises

Problem 1 (*EOQ* with positive lead time, shortages, production lot size): A retailer faces an annual demand for 2,400,000 shirts with the "Mumbo Jumbo Man–Savior of the Universe" logo. It costs \$450 to place a single order and the costs for keeping a single shirt in stock for an entire year are 60¢.

(a) How many units should the retailer order each time an order is placed and what are the associated costs?
(b) Given the result under (a), how many orders should be placed and what is the time between two consecutive orders (given a 360-day year)? What is the reorder point if the lead time were 20 days?
(c) Assume now that it is possible to allow shortages, given that the portion of the demand that cannot immediately be satisfied will be satisfied immediately after the next shipment arrives. It has been estimated that the associated loss of goodwill equals costs of 80¢ per shirt and year. Compute the order quantity, the maximum shortage, and the associated costs.
(d) What would happen to the results in (c) if the unit shortage costs were to increase by, say, 10¢? Explain in one short sentence, indicating the reason why. Calculations are not required.
(e) Suppose now that the retailer were to purchase the equipment to make the shirts in-house. The machine is capable of making 10,000 shirts per day (again based on a 360-day year). What is the number of shirts made in each production run? What are the total costs?
(f) How would the results under (e) change if the capacity of the machine under (e) were not 10,000 units per day but only 6,000? Explain in one short sentence.

Solution: (a) D=2,400,000, C_o = 450, C_h = 0.6, so that Q^* = 60,000 & TC^* = \$36,000.

(b) Then N^* = 40 & t_c^* = 1/40 [years] = 9 [days]. Reorder point R = $20(2{,}400{,}000/360) - \left\lfloor \frac{20}{9} \right\rfloor 60{,}000$ = \$13,333.33.

(c) Q^* = 79,372.54 and S^* = 34,016.80. Costs TC = (holding costs) + (ordering costs) + (shortage costs) = 7,775.27 + 13,606.72 + 5,831.45 = \$27,213.44.

(d) If c_s increases, shortages become more expensive, so that the order quantity Q^* and the maximum shortage S^* both decrease, while the total costs will increase.

(e) The regularity condition is satisfied. $Q^* = 103{,}923.05$ and $TC^* = 10{,}392.30 + 10{,}392.30 = \$20{,}784.60$.
(f) The regularity condition is violated, i.e., the machine capacity is insufficient to satisfy the demand.

Problem 2 (*EOQ*, positive lead time, shortages, production lot sizing): A retailer faces an annual demand for 4,000,000 pairs of sneakers with the "King Bong" logo. It costs \$100 to place a single order and the costs for keeping a single pair of sneakers in stock for an entire year are 50¢.

(a) How many pairs of sneakers should the retailer order each time an order is placed and what are the associated costs?
(b) Given the result under (a), how many orders should be placed and what is the time between two consecutive orders (given a 250-day year)? What is the reorder point if the lead time were 8 days?
(c) Assume now that it is possible to allow shortages, given that the portion of the demand that cannot immediately be satisfied will be satisfied immediately after the next shipment arrives. It has been estimated that the associated loss of goodwill equals costs of 20¢ per pair of sneakers and year. Compute the order quantity, the maximum shortage, and the associated costs.
(d) What would happen to the results in (c) if the unit shortage costs were to decrease by, say, 5¢? Explain in one short sentence, indicating the reason why. Calculations are not required.
(e) Suppose now that the retailer were to purchase the equipment to make the sneakers in-house. The machine is capable of making 20,000 pairs per day (again based on a 250-day year). How many pairs of sneakers are made in each production run? What are the total costs?
(f) How would the results under (e) change if the capacity of the machine under (e) were not 20,000 units per day but 25,000? Explain in one short sentence. No calculations are necessary.

Solution: (a) $D = 4{,}000{,}000$, $C_o = 100$, $C_h = 0.5$, so that $Q^* = 40{,}000$ and $TC^* = \$20{,}000$.

(b) Then $N^* = 100$ and $t_c^* = 1/100$ [years] $= 2.5$ [days]. Reorder point $R^* = 8(4{,}000{,}000/250) - \left\lfloor \frac{8}{2.5} \right\rfloor 40{,}000 = 8{,}000$.

(c) $Q^* = 74{,}833.15$ and $S^* = 53{,}452.25$. Costs TC = (holding costs) + (ordering costs) + (shortage costs) = 1,527.21 + 5,345.22 + 3,818.02 = \$10,690.45.
(d) If c_s decreases, shortages become even cheaper, so that the order quantity Q^* and the maximum shortage S^* both increase, while the total costs will decrease.
(e) The regularity condition is satisfied. $Q^* = 89{,}442.72$ and $TC^* = 4{,}472.14 + 4{,}472.14 = \$8{,}944.28$.
(f) The order quantity will decrease and the inventory-related costs will increase.

Problem 3 (quantity discounts): The annual demand for a product is 9,000, and the unit price of the product is \$5. The holding costs are estimated to be 10% of the price of the product, and the costs of placing a single order are \$1,000.

(a) Calculate the optimal order quantity and the associated costs.
(b) The supplier now offers a discounted price of \$4.80 if we order at least 8,000 units at a time. Should we take the offer? What are the associated costs?
(c) Our supplier adds another offer: a price of \$4.75, if we purchase at least 18,000 units (so that we have to order only every other year). Should we take this offer?

Solution: (a) With a unit price of p_0 = \$5, we obtain $Q^* = \sqrt{\frac{2(9{,}000)(1{,}000)}{(.1)(5)}} = 6{,}000$ with TC^* = 1,500 + 1,500 + 45,000 = \$48,000.

(b) The discounted price of p_1 = \$4.80 leads to an optimal quantity of Q^* = 6,123.72. This quantity does not qualify for a discount, so that we set $Q := Q_1$ = 8,000. The associated costs are $TC(Q_1)$ = 1,125 + 1,920 + 43,200 = \$46,245. As the total costs are lower at this price level, we should take the discount, order 8,000 units at a time, which will cost us \$46,245.

(c) The deeply discounted price of p_2 = \$4.75 leads to an optimal order quantity of Q^* = 6,155.87, not enough to qualify for the price level. As a result, we must increase the order quantity to the smallest level that allows the discount, i.e., $Q = Q_2$ = 18,000. At this level, the total costs are $TC(Q_2)$ = 500 + 4,275 + 42,750 = \$47,525, less than paying the full price, but not as good as the smaller discount determined under (b). Hence, the overall best option is to order 8,000 units each time we place an order, a policy that will cost us \$46,245 in each cycle.

Problem 4 (*EOQ* with stochastic lead time demand): Let the annual demand for a certain item be 1,000 units in the planning period. The holding costs are \$5 per unit, and the cost of placing an order is \$36 per order. The penalty cost for stockouts is \$3 per unit and stockout. There is a demand for 55 units during the lead time with a probability of 20%. The demand is 60 with a probability of 0.4, it is 65 with a probability of 0.30, and it is 70 with a probability of 0.10.

(a) Calculate the expected demand during lead time.
(b) Determine the economic order quantity and the resulting reorder point.
(c) Find the order quantity and the reorder point by simultaneous computation.
(d) What is the buffer stock?
(e) What is the minimal expected total or ordering, holding, and penalty costs?

Solution: (a) $E(d_L) = (55)(0.2) + (60)(0.4) + (65)(0.3) + (70)(0.1) = 61.5$.

(b) $Q_{EOQ} = \sqrt{\frac{2(1{,}000)36}{5}} = 120$ units, and with $F(R^*) \geq 1 - \frac{5(120)}{3(1{,}000)} = 1 - 0.2 = 0.8$, so that $R^* = 65$.

(c) Revising $Q^* = \sqrt{\frac{2(1{,}000)}{5}[36 + 3(5)0.1]} \approx 122.47$ units, $F(R^*) \geq 1 - \frac{5(122.47)}{3(1{,}000)} \approx 0.7959$, so that $R^* = 65$ units.

(d) The buffer stock is $R^* - \bar{d}_L = 65 - 61.5 = 3.5$ units.

(e) Costs TC_2 = (ordering costs) + (holding costs) + (penalty costs) = 293.95 + 323.68 + 12.25 = \$629.87.

11 Stochastic Processes and Markov Chains

Some of the previous chapter have dealt with random events. This chapter will deal with such events in a systematic way. In general, in stochastic processes, events occur over time. Time can be dealt with either in continuous fashion, or in discrete fashion. In the continuous case, we may look at the speed of an automobile at any given point in time or at the inventory level of a product in a supermarket at any time. In the discrete case, speed or inventory level are observed only during specific points in time, e.g., each minute, once a week or at similar intervals. In this chapter, we only deal with discrete-time models. The following three sections will introduce some of the basic ideas of stochastic processes and Markov chains.

11.1 Basic Ideas and Concepts

Consider first the random events that take place. Using the same examples as above, there is an infinite number of different speeds that a vehicle could be moving at, while the demand for a product may be very large, but is hardly infinite. Some types of events are much more restrictive: as an example, consider a light bulb. It will always be in exactly one of two "states of nature," in that it either works, or it does not. This is referred to as the *state space*, i.e., the number of different states the "system" can possibly be in. As already hinted at, the individual states are similar to the states of nature in decision analysis, see Chapter 9 in this volume. This chapter deals only with processes that have a finite state space.

Each event in this *discrete-time, finite state space process* is then a random variable X_t that depends on the time t at which it is observed. As an illustrative example, consider a used car. Lately, the vehicle has displayed the warning message "service engine soon" and it is known that this means that the vehicle is in one of four states: it either runs well (state s_1), it runs with minor problems (state s_2), it runs with major problems (state s_3), or it fails altogether (state s_4). At any point in time, the vehicle is in exactly one of these four states. It stands to reason that the state that the vehicle is in one year does depend on the state the car

H.A. Eiselt and C.-L. Sandblom, *Operations Research: A Model-Based Approach*,
DOI 10.1007/978-3-642-10326-1_11, © Springer-Verlag Berlin Heidelberg 2010

was in the year before. More specifically, we can define *transition probabilities* p_{ij} that indicate that the vehicle is in state j, given that it was in state i in the previous year. (We assume that no repairs are performed). It is apparent that the transition probabilities are conditional probabilities of the type $p_{ij} = P(X_{t+1} = j | X_t = i)$, or in simple words, the probability that the random variable is in state s_j in year t+1, given that it was in state s_i in year t. As a numerical illustration, consider the matrix $\mathbf{P} = (p_{ij})$ of transition probabilities

$$\mathbf{P} = \begin{bmatrix} 0.90 & 0.06 & 0.03 & 0.01 \\ 0 & 0.85 & 0.10 & 0.05 \\ 0 & 0 & 0.75 & 0.25 \\ 0 & 0 & 0 & 1.00 \end{bmatrix}$$

For example, if the vehicle has been running well in the previous year, then there is a 90% chance that it will be running smoothly in this year as well, as shown by the element p_{11}. Or, if the vehicle experiences minor problems this year, then there is a 5% chance that it will fail altogether next year, as shown by the element p_{24}.

A few features of this transition matrix are noteworthy. First of all, note that all elements below the main diagonal (i.e., in the lower left corner) are zero. This simply means that, given that no repairs are performed, the vehicle will never improve. Secondly, we can observe that the sum of probabilities in each row of the transition matrix equals one. This is always the case, as the transition probabilities are conditional probabilities, and as such, given that we are in a row at time t, we *must* choose a successor state for time t+1. Thirdly, notice that once the car is in state s_4 (i.e., the vehicle fails), then it will never get out of it again. Thus, the state s_4 is referred to as an *absorbing state*. The example presented here is a stochastic process with the *Markovian property* that holds, if the present state of the process depends only on the state of the system immediately prior to this and the transition probabilities. (As a historical aside, Andrey Andreyevitch Markov, 1856-1922, was a Russian mathematician who made important contributions to the field of probability and statistics).

A nice visual representation is the *transition diagram*. In this diagram, the nodes represent the states of the process, and the arcs represent transitions with a positive probability. The transition diagram for our automobile example is shown in Figure 11.1.

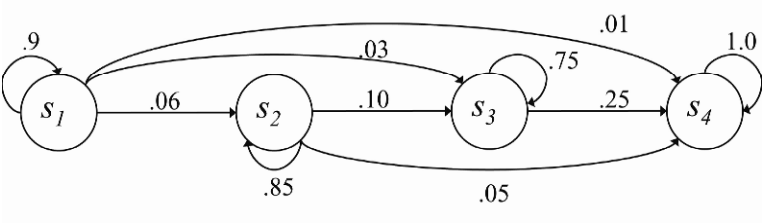


Figure 11.1

Note that absorbing states have only arcs leading into them, but not out (at least not to other states). Here, we also assume that the process is *stationary*, meaning that the transition probabilities do not change over time.

An obvious question to be asked is what happens, if we go through more than one transition. In other words, the transition probabilities tell us what the likelihood is to be in one state one period after the process starts. But what about two or more periods? This is the question we discuss in the next paragraphs. In order to facilitate the discussion, consider a simple example involving stock prices. In particular, we only allow an upward movement of the price and a downward movement. The transition probabilities are shown in the matrix

$$\mathbf{P} = \begin{bmatrix} 0.2 & 0.8 \\ 0.7 & 0.3 \end{bmatrix}.$$

In other words, if the stock went up today, then there is a 20% chance that it will go up again tomorrow, while there is an 80% chance that the stock's price will decline. Similarly, if the price of the stock decreased today, then there is a 70% chance that it will increase tomorrow and a 30% chance that it will decrease tomorrow. When analyzing changes after multiple periods, we could use a *time-state graph* as shown in Figure 11.2, where the nodes s_i^t indicate the state of nature i at the end of period t, the arcs denote the possible transitions, and their values are the transition probabilities. As the transition probabilities are stationary, an arc from, say, s_i^t to s_j^{t+1} will have the same value as, say an arc from s_i^{t+2} to s_j^{t+3}.

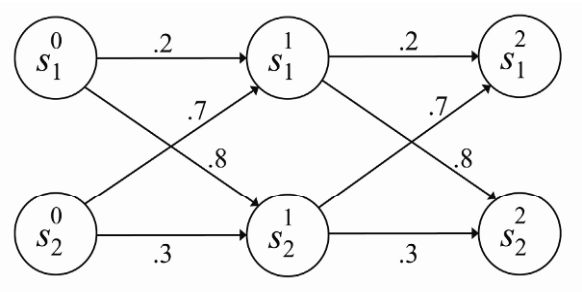


Figure 11.2

Considering Figure 11.2, assume now that we are presently in an upswing of the stock price, i.e., in state s_1^0. The probability that there will be an increase in stock prices two days later will have to examine all paths from the present state s_1^0 to the state s_1^2. Here, there are exactly two paths: the first path leads from s_1^0 to s_1^1 and on to s_1^2, while the second path leads from s_1^0 first to s_2^1 and then to s_1^2. Note that the first path considers a price increase on the first day, while the second has a decrease on the first day. The probability is then calculated as the sum of probabilities along each path, which, in turn, are computed as the product of all transition probabilities along the path. In this example, the probability on the former path s_1^0, s_1^1, s_1^2 is (.2)(.2) = .04 (meaning that there is only a 4% chance of two price increases in a row), while the latter path s_1^0, s_2^1, s_1^2 has a probability of (.8)(.7) = .56, meaning that the probability of a price decrease on day one, followed by a price increase on day two has a probability of 56%. This means that we obtain the transition probability that that state changes from a price increase on day t to a price increase on day t+2 as 0.04 + 0.56 = 0.60. Similarly, we compute the remaining probabilities, resulting in the matrix $\mathbf{P}^2$, which indicates the transition probabilities from day t to day t+2. It is

$$\mathbf{P}^2 = \begin{bmatrix} 0.6 & 0.4 \\ 0.35 & 0.65 \end{bmatrix}.$$

This procedure can be repeated for any number of days. While this is possible, the procedure is extremely awkward. We can achieve the same results by simple matrix multiplication. In particular, we can obtain $\mathbf{P}^2 = \mathbf{PP}$, $\mathbf{P}^3 = \mathbf{P}^2\mathbf{P} = \mathbf{PPP}$, and so forth, so that we obtain $\mathbf{P}^r = \mathbf{PPP}\ldots\mathbf{P}$, i.e., the matrix $\mathbf{P}$ multiplied itself r times. For instance, in the stock example we obtain

$$\mathbf{P}^3 = \mathbf{P}^2\mathbf{P} = \begin{bmatrix} 0.6 & 0.4 \\ 0.35 & 0.65 \end{bmatrix}\begin{bmatrix} 0.2 & 0.8 \\ 0.7 & 0.3 \end{bmatrix} = \begin{bmatrix} 0.4 & 0.6 \\ 0.525 & 0.475 \end{bmatrix}.$$

This means that given that there has been an increase in prices on day one, then there is a 40% chance that there will be a rise in prices again two days later, and similar for the other elements of the matrix.

So far, we have only dealt with conditional probabilities, i.e., *given* that a certain state of nature prevails in the beginning, we computed the probability that some (other) state of nature occurs after a given number of periods. Below, we start the process with an initial probability distribution that assigns an unconditional probability u_i^0 to each state of nature s_i. These probabilities are then collected in the initial probability row vector $\mathbf{u}^0 = [u_1^0, u_2^0, ..., u_m^0]$, assuming that there is a total of m states of nature. The unconditional probabilities that describes the likelihood that the system is in of the states of nature after one week is $\mathbf{u}^1$, which can be computed by simple vector-matrix multiplication as $\mathbf{u}^1 = \mathbf{u}^0\mathbf{P}$. In general, we have

$$\mathbf{u}^r = \mathbf{u}^{r-1}\mathbf{P} = \mathbf{u}^0\mathbf{P}^r.$$

As an illustration, consider again the car example. Recall that its single-stage transition matrix was

$$\mathbf{P} = \begin{bmatrix} 0.90 & 0.06 & 0.03 & 0.01 \\ 0 & 0.85 & 0.10 & 0.05 \\ 0 & 0 & 0.75 & 0.25 \\ 0 & 0 & 0 & 1.00 \end{bmatrix}$$

with the states of nature being s_1: running well, s_2: running with minor problems, s_3: running with major problems, and s_4: not running. Suppose now that the car is initially running with minor problems, i.e., $\mathbf{u}^0 = [0, 1, 0, 0]$. After one year, we have the probabilities

$$\mathbf{u}^1 = \mathbf{u}^0\mathbf{P} = [0, 1, 0, 0] \begin{bmatrix} 0.90 & 0.06 & 0.03 & 0.01 \\ 0 & 0.85 & 0.10 & 0.05 \\ 0 & 0 & 0.75 & 0.25 \\ 0 & 0 & 0 & 1.00 \end{bmatrix} = [0, 0.85, 0.10, 0.05],$$

i.e., there is no possibility that the vehicle with perform perfectly, there is an 85% chance that it will continue to run with minor problems, there is a 10% chance that the problems are now major, and there is a 5% chance that the vehicle will fail altogether. For the second year, we obtain

$$\mathbf{u}^2 = \mathbf{u}^1\mathbf{P} = [0, 0.85, 0.10, 0.05] \begin{bmatrix} 0.90 & 0.06 & 0.03 & 0.01 \\ 0 & 0.85 & 0.10 & 0.05 \\ 0 & 0 & 0.75 & 0.25 \\ 0 & 0 & 0 & 1.00 \end{bmatrix} = [0, 0.7225, 0.16, 0.1175],$$

indicating that there is now a 72.25% chance that the minor problems will persist, a 16% chance that the problems are now major, and an 11.75% chance that the vehicle will fail.

11.2 Steady-State Solutions

One question is what will happen, if the process will somehow converge after a large number of transitions. If it does, we will call the resulting solution a *steady-state solution*. Clearly, a steady-state solution is an ideal concept that is not very likely going to be realized in practice: take, for instance the used car example. While we may keep the vehicle for a long time, the time will be finite, while a steady state, a state that no longer depends on the initial conditions, it is typically reached only after an infinite number of transitions. Still, the steady-state is an important concept that will tell us to what a process converges, provided that it converges at all. A sufficient condition for the existence of a steady state is given if each state can be reached from each other state on a path that has positive probability, and if there exists at least one state that leads to itself with a positive probability.

We showed in the previous section that $u^n = \mathbf{u}^0\mathbf{P} = \mathbf{u}^{n-1}\mathbf{P}$, and if n tends to infinity, we obtain $\mathbf{u}^\infty = \mathbf{u}^\infty\mathbf{P}$. To distinguish the steady-state solutions from all others, it is customary to replace $\mathbf{u}^\infty$ by $\boldsymbol{\pi}$, so that the steady-state solutions will satisfy $\boldsymbol{\pi} = \boldsymbol{\pi}\mathbf{P}$. In addition, we have to ensure that the sum of all elements in $\boldsymbol{\pi}$ equals 1, as all components of $\boldsymbol{\pi}$ are probabilities that are mutually exclusive and collectively exhaustive.

In the used car example, the system of simultaneous linear equations is

$$\begin{aligned} \pi_1 &= 0.9\pi_1 \\ \pi_2 &= 0.06\pi_1 + 0.85\pi_2 \\ \pi_3 &= 0.03\pi_1 + 0.10\pi_2 + 0.75\pi_3 \\ \pi_4 &= 0.01\pi_1 + 0.05\pi_2 + 0.25\pi_3 + \pi_4 \\ \pi_1 + \pi_2 + \pi_3 + \pi_4 &= 1 \end{aligned}$$

The first equation requires that $\pi_1 = 0$, inserting this result in the second equation leads to $\pi_2 = 0$, and using this result in the third equation leads to $\pi_3 = 0$. The fourth equation then reduces to the tautological identity $\pi_4 = \pi_4$, but the last

equation then helps to solve the system with $\pi_4 = 1$. This is the obvious result mentioned earlier in this section.

The stock example provides another illustration of the concept. It is easily seen that the Markov chain is ergodic, so that a steady-state is certain to exist. In order to determine the (unconditional) steady-state probabilities, we solve the system

$$[\pi_1, \pi_2] = [\pi_1, \pi_2]\begin{bmatrix} 0.2 & 0.8 \\ 0.7 & 0.3 \end{bmatrix},$$

which can be written as

$\pi_1 = 0.2\pi_1 + 0.7\pi_2$
$\pi_2 = 0.8\pi_1 + 0.3\pi_2$, coupled with
$\pi_1 + \pi_2 = 1$.

The system has steady-state probabilities $\pi_1 = 7/15$ and $\pi_2 = 8/15$, or $\pi \cong [0.4667, 0.5333]$. In other words, in the long run we can expect stock prices to rise about 47% of the time, while we can expect them to drop about 53% of the time.

11.3 Decision Making with Markov Chains

This section demonstrates how we can use the results obtained in the previous two sections in the context of decision making. To explain, consider again the used car example of the previous sections. Suppose now that there are three automobiles for sale . They are of the same make and model and the transition probabilities in the matrix **P** above apply to all of them. One vehicle is in perfect running order, and it sells for \$5,000. The second vehicle runs with minor problems and it sells for \$4,000, while the third vehicle has major problems and it costs \$2,000. After two years we want to sell the vehicle again. Its price then (as it does now) will depend on its state. In particular, it has been estimated that a vehicle in perfect condition will sell for \$3,500, one with minor problems sells for \$2,500, and a car with major problems sells for \$500. We assume that there is no difference between the cars in maintenance costs and that no repairs are made (which is not really realistic). Which car should we purchase?

We can address the problem by considering each option, one at a time. If we purchase the vehicle that is in perfect condition, then we decide that $\mathbf{u}^0 = [1, 0, 0, 0]$. After one year, we obtain $\mathbf{u}^1 = [0.9, 0.06, 0.03, 0.01]$ and after two years we have $\mathbf{u}^2 = [0.81, 0.105, 0.0555, 0.0295]$. In other words, when we attempt to sell the vehicle, it will still be in perfect shape with a probability of 81%, it will have minor problems with a probability of 10.5%, and so forth. The expected price we can sell the vehicle for is then 3,500(0.81) + 2,500(0.105) + 500(0.0555) + 0(0.0295) = \$3,125.25, resulting in a loss of 5,000 – 3,125.25 = \$1,874.75.

The other two vehicles are dealt with similarly. If we purchase the car with minor flaws, we decide to choose $\mathbf{u}^0 = [0, 1, 0, 0]$ and obtain the probability vectors $\mathbf{u}^1 = [0, 0.85, 0.10, 0.05]$ after one year and $\mathbf{u}^2 = [0, 0.7225, 0.16, 0.1175]$ after two years of ownership, so that the expected price of the car at the time of sale is \$1,886.25, which means a loss of \$2,113.75.

Finally, consider the car with major flaws. Deciding to purchase it in the beginning means to set $\mathbf{u}^0 = [0, 0, 1, 0]$, so that by the end of year 1, we have $\mathbf{u}^1 = [0, 0, 0.75, 0.25]$, and by the end of the second year (or, equivalently, at the beginning of year 3), we have probabilities $\mathbf{u}^2 = [0, 0, 0.5625, 0.4375]$, so that the expected price of sale is \$281.25 for an expected loss of \$1,718.75. Given our assumptions, our best bet would be to purchase the car with the major flaws.

Consider now the possibility of either purchasing a warranty repair policy for \$100 per month or pay the \$1,800 repair bill, whenever state s_4 occurs. In either case, whenever the system enters state s_4, (the car fails), it stays in this state for one period during which it is repaired and, at the end of this period, returns to s_1 (runs without problems). The transition matrix then changes to

$$\mathbf{P} = \begin{bmatrix} 0.90 & 0.06 & 0.03 & 0.01 \\ 0 & 0.85 & 0.10 & 0.05 \\ 0 & 0 & 0.75 & 0.25 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

As opposed to the previous case without repairs, this transition matrix does not have an absorbing state. Finding the steady-state probabilities for this case, we obtain $\pi = [\pi_1, \pi_2, \pi_3, \pi_4] = [\frac{50}{89}, \frac{20}{89}, \frac{14}{89}, \frac{5}{89}] \approx [.5618, .2247, .1573, .0562]$. As above, we can purchase a vehicle in state s_1, s_2, and s_3 for \$5,000, \$4,000, and \$2,000, respectively, while we now add that a failed vehicle can be purchased for \$1,000. (Earlier, it made no sense to purchase a vehicle in state s_4, as there were no repairs). This means that in the long run, the annual expected cost of repairing the vehicle is 0.0562(1,800) = \$101.16 if we pay ourselves, or \$100 for the policy, which makes the purchase of the policy slightly superior.

Consider now again the case, in which four vehicles of the same make and model are offered, one in each of the four states. The purchase prices and the repair costs are the same as above. The idea is to purchase the vehicle at the end of year 0 (or, equivalently, the beginning of year 1) and sell it again at the end of year 3 (the beginning of year 4). The price for which the vehicle can be sold at that time has been estimated to be \$2,500 if it is running perfectly (state s_1), \$1,500 if it runs with minor problems (state s_2), \$800 in case it has major problems (state s_3), and \$100 if it fails (state s_4). Analyzing this case necessitates the use of solution trees

rather than the computation of the states $\mathbf{u}^0$, $\mathbf{u}^1$, $\mathbf{u}^2$, and $\mathbf{u}^3$. The reason is that we can, for instance, purchase a vehicle with major flaws today and sell it at the end of year 3 with major flaws. However, it may happen that this vehicle simply stayed in this state, or it failed, we repaired it and it was perfect for one year, and then it degenerated again to the state with major problems. The two cases have the same initial state and the same state at the end of the planning period, but they incur very different costs.

In particular, we will need four probability trees, one for each vehicle we can purchase. For reasons of limitations of space, we display only the solution tree for the vehicle that is initially running with major problems. The different states are abbreviated as *P* (perfect running condition, state s_1) *Mi* (state s_2, minor problems), *Ma* (state s_3, major problems) and *F* (state s_4, fail). All possible sequences of states that may occur are shown in Figure 11.3. The two numbers next to the last set of nodes when we sell the vehicle denote the probability that this string of events occurs, and the costs occurred throughout the time that we own the vehicle. The latter includes the purchase price plus repairs, if any, minus the price we obtain when we sell the car. As an example, consider the string of events that can be described as *P* – *F* – *P* – *Mi*. In other words, the car runs perfectly when we purchase it, it then fails in the next year, after which it runs perfectly again, and it exhibits minor problems after that. The probability for such a sequence of events is (0.01)(1)(0.06) = 0.0006, and the costs include the purchase price of $5,000, the repair bill of $1,800, and the sales price of $1,500, resulting in overall costs of $5,300. The expected costs for purchasing a vehicle in perfect condition and keeping it for three years are then $2,888.42.

We can then perform a similar analysis with a vehicle that is initially in state s_2 (running with minor problems). The tree is a bit smaller, and the expected costs are $3,650.21. Purchasing a car that has major problems will cost $1,375.92, while buying a vehicle that fails and must be repaired right away will have expected costs of $564.15, making this the preferred strategy of the buyer.

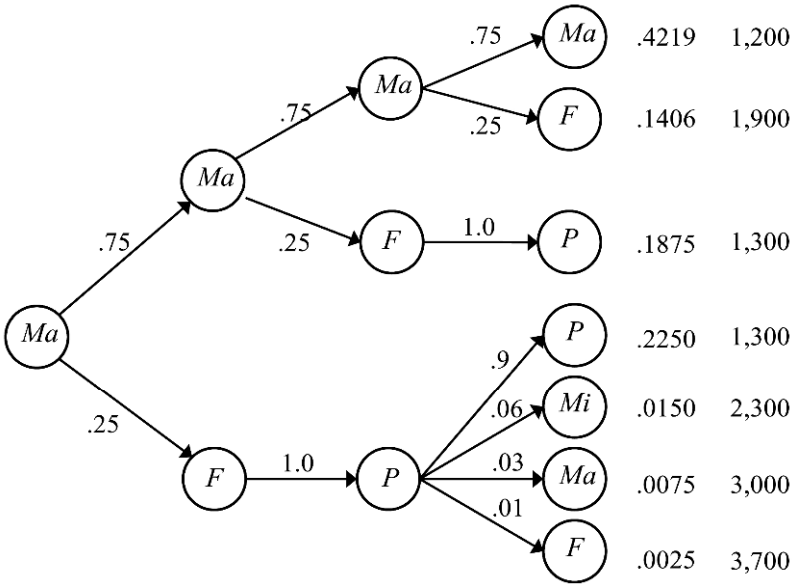


Figure 11.3

Exercises

Problem 1 (brand switching, computation of steady-state probabilities): Three major motel chains compete for the lucrative economy market: Sunny 6, Cloudy 7, and Rainy 8. It has been observed that customers who stay in one of the motels usually patronize the same chain again, except if they perceive the service to be poor. In particular, in case of Sunny 6, 10 percent of the time the service is perceived to be poor in which case customers change to one of the other two chains with equal probability. In case of Cloudy 7, service is perceived to be satisfactory 80 percent of the time; if it is not, customers switch to one of the other two chains with equal probability. Finally, 90 percent of the time service at Rainy 8 is deemed to be ok; if it is not, customers always switch to Sunny 6.

(a) Set up the transition matrix **P**.
(b) In the long run, what percentage of customers patronize the three motels?
(c) Management of the Cloudy 7 chain perceives that its customer loyalty is not as good as one could wish. By using of better management techniques, they may be able to reduce the proportion of poor service to ten percent (again, in case of poor service, customers switch to the other chains with equal likelihood). What are the new steady-state proportions?
(d) Given that one percent of business is worth $500,000, what is the maximum amount that the management of Cloudy 7 should be prepared to pay to implement the new management techniques that result in the improved service?

Solution: (a) The transition matrix is

$$\mathbf{P} = \begin{bmatrix} .9 & .05 & .05 \\ .1 & .8 & .1 \\ .1 & 0 & .9 \end{bmatrix}.$$

(b) The steady-state probabilities are $\boldsymbol{\pi} = \left[\frac{4}{8}, \frac{1}{8}, \frac{3}{8}\right] = [0.5000, 0.1250, 0.3750]$.

(c) The revised transition matrix is

$$\mathbf{P} = \begin{bmatrix} .9 & .05 & .05 \\ .05 & .9 & .05 \\ .1 & 0 & .9 \end{bmatrix}.$$

The steady-state probabilities are $\boldsymbol{\pi} = \left[\frac{4}{9}, \frac{2}{9}, \frac{3}{9}\right] = [0.4444, 0.2222, 0.3333]$.

(d) The market share of the Cloudy 7 chain has increased by 9.7222%, which is worth \$4,861,100, which is the maximal amount the management of Cloudy 7 should pay for the change.

Problem 2 (criminal recidivism, value of a policy): Consider a city of 500,000 people, which have been classified by the authorities as either "not criminal," "misdemeanor," or "felon." Long-term studies indicate that the transition of an individual from one state to another from one year to the next is shown in the transition matrix

$$\mathbf{P} = \begin{bmatrix} 0.95 & 0.04 & 0.01 \\ 0.50 & 0.40 & 0.10 \\ 0.10 & 0.30 & 0.60 \end{bmatrix}.$$

City council contemplates a new program that costs \$70,000,000 and changes the transition probabilities to

$$\mathbf{P'} = \begin{bmatrix} 0.95 & 0.04 & 0.01 \\ 0.70 & 0.25 & 0.05 \\ 0.20 & 0.40 & 0.40 \end{bmatrix}.$$

Assume that an individual in the "misdemeanor" category costs about \$1,000 annually, while the cost of a felon are \$10,000 each year.

(a) What is the probability that a felon does not commit any crime in the next two years with and without the crime prevention initiative?

(b) What is the probability that an individual in the "misdemeanor" category does not commit any crime in the next three years with and without the crime prevention initiative?
(c) What are the steady state probabilities with and without the initiative? Compare the long-term costs with and without the initiative.

Solution: (a) The probability of the string "felon" – "no crime" – "no crime" is (0.1)(0.95) = 0.095 without the initiative and (0.2)(0.95) = 0.19 with it.
(b) The probability of the string "misdemeanor" – "no crime" – "no crime" – "no crime" is (0.5)(0.95)(0.95) = 0.45125 without and (0.7)(0.95)(0.95) = 0.63175 with it.
(c) The steady-state probabilities are $\boldsymbol{\pi} = [\pi_1, \pi_2, \pi_3] = \left[{}^{210}\!/_{239}, {}^{19}\!/_{239}, {}^{10}\!/_{239}\right] \approx [0.87866, 0.07950, 0.04184]$ without the initiative and $[\pi_1, \pi_2, \pi_3] = \left[{}^{860}\!/_{935}, {}^{56}\!/_{935}, {}^{19}\!/_{935}\right] \approx [0.91979, 0.05989, 0.02032]$. Without the initiative, the expected costs per individual are \$497.908, while they are \$263.102 with it. For the city of 500,000, this means costs of \$248,954,000 without the initiative and \$131,551,000 with it, a savings of \$117,403,000. Since the savings exceed the costs of \$70,000,000, the initiative should be introduced.

12 Waiting Line Models

Waiting line or queuing system are pervasive. Many of us remember the long lineups in front of stores in the Soviet Union and Vietnam, and we have all experienced lineups in banks and supermarkets, but there are many more instances with waiting lines: think, for instance, about traffic lights, where drivers line up and wait, files that wait for processing in the inbox at a clerk's workstation, or telephone calls that are put in a queue. Queuing system were first examined by Agner Krarup Erlang (1878–1929). Erlang was a Danish mathematician, who worked for the Copenhagen Telephone Company. One of the questions he faced during this time was to determine the number of telephone circuits that are required to provide an acceptable level of service.

A.K. Erlang

Phone switchboard operator

H.A. Eiselt and C.-L. Sandblom, *Operations Research: A Model-Based Approach*,
DOI 10.1007/978-3-642-10326-1_12, © Springer-Verlag Berlin Heidelberg 2010

12.1 Basic Queuing Models

In order to avoid discussing only special cases, we will formalize queuing systems as follows. All entities that are in need of service of some kind will be referred to as *customers*, while the service is performed at *service stations*. The process can then be thought of as follows. Customers are in a *calling population*. Once they are in need of service (here, for simplicity, we will consider only a single type of service customers are interested in), they will approach the service system, where they will line up. When it is the customer's turn, he will be served, after which the customer will leave the service system and rejoin the calling population. The structure of this process can be visualized in Figure 12.1.

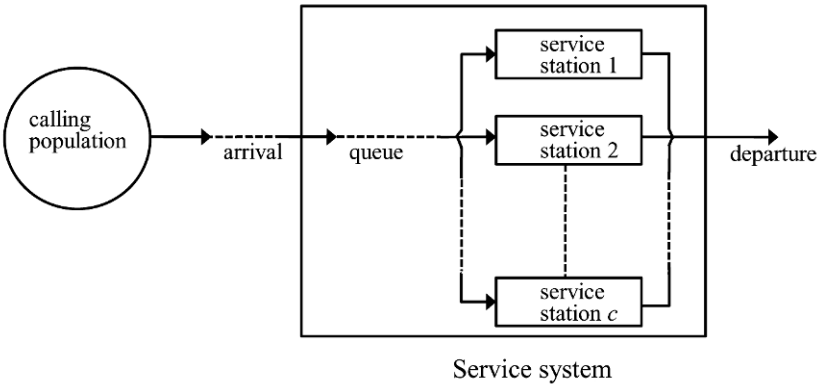


Figure 12.1

The service system will require some further specifications. First of all, each service system has only a single waiting line. A service system typically consists of a number of c parallel service stations, each assumed to perform the same type of service. Parallel service stations are usually referred to as *channels*. In some instances, one channel consists of a series of service stations: imagine entering a building, where a potential customer first have to be cleared by security, then being directed to a general secretary, from where the service continues to the department director's secretary, and finally on to the department director. At each station, the customer may be asked to leave the system, e.g., for not clearing security, the unavailability of a service station, and for other reasons. Multi-phase systems can be very complex and will not be discussed here.

In order to categorize queuing systems, Kendall (1918-2007) devised a taxonomy in 1953 that, in a variety of versions, is the standard to this day. The original system consists of three descriptors, and it has been extended to at most six components.

In order to be as general as possible, we introduce the complete six-component system first, but then use only the more compact 3-component system later on. The notation is

$$A/B/C \quad K/N/D,$$

where each letter is a placeholder for one component of waiting lines. These components are now described in some detail.

The symbol *A* describes the arrival process. We will use the symbol "*M*," if arrivals are random and follow a so-called Poisson process (where the "*M*" stands for "Markovian)." The Poisson distribution (see Appendix D in this book) describes such a process. Other popular processes include "*D*" or deterministic arrivals (in which the time between arrivals is known with certainty, as is the case on an assembly line) or "*G*," where arrivals follow some general distribution, for which only some key parameters are known, e.g., mean and variance.

The second component "*B*" symbolizes the service process. Again, the letter "*M*" symbolizes Markovian, i.e., exponential service time, while "*D*" and "*G*" (or, more precisely, "*GI*") symbolize deterministic and general (general independent) service times, respectively.

The letter *C* indicates the number of parallel service stations.

The letter *K* in the extended version of the taxonomy describes the number of customers that can be accommodated in the entire system. This includes the number of customers who wait as well as those that can be served. In case this descriptor is not used, it is assumed that *K* is infinite. While no real-life system has infinite capacity, assuming an infinite *K* simplifies the analysis tremendously and is usually very close to the true results as soon as *K* exceeds 30 or 40.

The symbol *N* denotes the size of the calling population. As in the case of the capacity of the system, *N* will assumed to approach infinity if it is not specified. Again, for reasonably large values of *N*, we may assume its value to be sufficiently close to infinity so as to simplify the computations.

Finally, the symbol *D* denotes the queuing discipline. Typical disciplines are *FCFS* or *FIFO* (first-come, first-served or first in, first out), *LCFS* or *LIFO* (last come, first served or last in, first out), or *SIRO* (service in random order). An important category are priority queues, in which some customers receive preferential treatment. The most prominent example of priority queues occurs in health care, where more serious cases will be treated first. In case no queuing discipline is specified, it is assumed that the *FCFS* discipline applies.

In general, it will be useful to think about waiting lines as buffers between arrival and service. A good image would be one of a water tank. While a waiting line

(similar to an inventory) grows with the inflow, it shrinks whenever service is provided and customers leave the system.

In queuing theory, we distinguish between transient states and steady states. While a *steady state* occurs if the system has been running for a very long time, *transient states* are still, at least to some degree, dependent on the initial state of the system. A simple example is the opening of a new cash register in a store. Initially, no one is in the system and waiting times will be short. However, as the cash register operates for some time, the service system becomes more congested and is no longer dependent on the opening conditions.

The key task of queuing theory is to compute measures of interest from some key input factors or queue characteristics. The key characteristics of a queue are those described in the taxonomy. Measures of interest include average waiting times, the probability that a newly arriving customer will have to wait, the average length of a queue, and others. In order to formalize our discussion, we use the following conventions about notation (which, incidentally are almost all standard in the pertinent literature):

λ denotes the mean *arrival rate*, measured in [customers/hour]. It is the average number of customers who *actually arrive* at the system in the specified amount of time.

μ is the mean *service rate*, measured in [customers/hour]. It is the average number of customers who *can be served* by a single service station.

It is worth noting that while λ expresses an actual observable fact, μ indicates a capability. The actual number of customers served does not only depend on the service station's capabilities, but also on the number of customers who desire service (which depends on λ).

The inverse values of λ and μ also have important interpretations. Suppose that a service station can deal with $\mu = 12$ customers per hour, then the inverse value is $1/\mu = 1/12$ [hours/customer] = 5 [minutes/customer]. This is the average *service time*. The inverse value of the arrival rate can be interpreted similarly. If the average arrival rate is $\lambda = 10$ [customers/hour], then $1/\lambda = 1/10$ [hours/customer] = 6 [minutes/customer], meaning that on average, six minutes elapse between two successive arrivals. This is referred to as the (average) *interarrival time*.

It is apparent that, in case of a single service station, the arrival rate cannot exceed the service rate. If it would, then there are more arrivals than can be handled by the service station, so that—given infinite patience of the customers—the waiting line will grow towards infinity. This gives rise to a regularity condition. In order to express it in a compact form, we define the *traffic intensity* (sometimes also

referred to as *utilization rate*) $\rho = \lambda/\mu$. A feasible system (i.e., a system that has a steady state) must then have $\rho < c$, where c denotes the number of parallel service stations. As an example, consider a system with a single service station that faces an arrival rate of $\lambda = 24$ customers per hour. For it to be feasible, the service rate must be $\mu > 24$, or, equivalently, the average service time cannot exceed 150 seconds.

On the output side are the measures that we are interested and that we can compute. They include:

P_n: the probability that there are n customers in the system (with the important special case of $n = 0$ and P_0, i.e., the probability that the system is idle),
W_s: the average waiting time per customer,
W_q: the average time a customer spends in the queue,
L_s: the average number of customers in the system, and
L_q: the average number of customers in the queue.

Before stating formulas for these measures, there are some general relations that hold in queuing, regardless of the specific system under consideration. The first such relation is

$$W_s = W_q + 1/\mu. \tag{1}$$

Simply stated, the relation expresses that the total time a customer spends in the system equals the waiting time plus the service time. Another relation is known as Little's formula (based on the work by J.D.C. Little who published the formula in 1961), which states that

$$L_\bullet = \lambda W_\bullet, \tag{2}$$

where the "•" stand for either the subscript "*s*" or the subscript "*q*". This formula provides a convenient way to compute the number of customers in the queue and in the system from the average time customers spends in the queue or in the system, respectively (or vice versa).

The simplest queuing model is the *M*/*M*/1 model, in which the number of customer arrivals are random and follow a Poisson process (making the interarrival times exponentially distributed), and the service time is exponential (so that the service rate again follows a Poisson distribution). The key queuing formulas for this model are shown in Table 12.1.

Table 12.1: Steady-state formulas for the *M*/*M*/1 queuing model

$P_0 = 1 - \rho$	$L_s = \frac{\lambda}{\mu - \lambda}$	$W_s = \frac{1}{\mu - \lambda}$
$P_n = P_0 \rho^n$	$L_q = \frac{\rho\lambda}{\mu - \lambda} = \frac{\rho^2}{1 - \rho}$	$W_q = \frac{\rho}{\mu - \lambda}$

As an illustration, consider the following

Example: Customers arrive at the counter of a bank at a rate of 30 per hour. Arrivals are random and service time is exponential, so that we are dealing with an *M*/*M*/1 model. The clerk's average service time is 90 seconds. Putting the parameters in their required form, we glean $\lambda = 30$ and $\mu = 45$ from this information. As $\rho = \lambda/\mu = 30/45 = ⅔ < 1$, the system does have a steady state. The probability that the bank teller is idle is $P_0 = 1 - \rho = ⅓$. The probability that at least two customers are waiting equals the probability that there are at least three customers in the bank or, formally, $P_3 + P_4 + P_5 + \ldots = 1 - P_0 - P_1 - P_2 = 1 - ⅓ - ⅓(⅔)^1 - ⅓(⅔)^2 \cong \frac{8}{27} \approx .2963$ or slightly less than one third. On average, there are $L_q = 1.3333$ customers waiting in line and the average time a customer spends in the system is $W_s = 1/15$ {hour] = 4 minutes.

An interesting case arises when the decision maker specifies the service level and determines bounds for the capabilities of the servers. Suppose that in an *M*/*M*/1 system with $\lambda = 20$, the decision maker specifies that the probability that there are three or more customers in the system should not exceed 95%. The probability of three or more customers in the system is again $1 - P_0 - P_1 - P_2 = 1 - (1-\rho) - \rho(1-\rho) - \rho^2(1-\rho) = 1 - 1 + \rho - \rho + \rho^2 - \rho^2 + \rho^3 = \rho^3$. As this probability should not exceed 95%, we obtain the condition $\rho^3 = \frac{\lambda^3}{\mu^3} \le .95$, or, as $\lambda = 20$, $\mu^3 > 8{,}000/.95$ or $\mu > 20.3449$. The reason that it is sufficient that the service rate barely exceeds the arrival rate is that the probability of three or more customers in the system is very small.

Suppose now that the service rate is no longer random but that it follows some general distribution. All we know about this distribution is that the mean service time is $1/\mu$ and the variance of the distribution equals σ^2. This means that we are dealing with an *M*/*G*/1 model, for which some rather elegant formulas are available. Again, as in all single channel systems in a steady state, $P_0 = 1 - \rho$. Furthermore, the *Pollaczek-Khintchine formula* developed in 1930 is

$$L_q = \frac{\lambda^2\sigma^2 + \rho^2}{2(1-\rho)}. \qquad (3)$$

The values of L_s, W_q, and W_s can then be computed based on the general relations (1) and (2). Before demonstrating this model on a numerical example, note that the $M/M/1$ model is a special case of the $M/G/1$ model with $\sigma^2 = 1/\mu^2$. Replacing σ^2 in the above expression results in $L_q = \frac{\rho^2}{1-\rho}$, which is the standard formula of the $M/M/1$ model. Similarly, we observe that in the case of the $M/D/1$ model, i.e., the queuing model with deterministic service time, the variance $\sigma^2 = 0$, so that the Pollaczek-Khintchine formula reduces in this case to $L_q = \frac{\rho^2}{2(1-\rho)}$. Observe that the number of customers waiting in the case of the deterministic model is exactly half of that of the standard model with exponential service time. In other words, the performance of the queuing system can be improved quite dramatically by reducing the variance of the service time.

Example: Arrivals of customers at a single service desk follow a Poisson distribution, while the service time follows a general distribution. There is an average of $\lambda = 15$ arrivals, while the service time is 3 minutes on average with a standard deviation of 6 minutes. This means that $1/\mu = 1/20$ [hours] and $\sigma^2 = 1/(10)^2 = 1/100$, so that the average number of customers waiting in line is then $L_q = 5.625$ and the average waiting time is $W_q = .375$ hours = 22.5 minutes. If the service time were exponential, we obtain the standard $M/M/1$ system and the performance measures $L_q = 2.25$ and $W_q = 9$ minutes, while the deterministic model has $L_q = 1.125$ and $W_q = 4.5$ minutes.

Consider now the case of multiple service stations. Here, we have to make the distinction between one multi-station service center and a number of parallel single-service centers. The general rule is that each service center has only a single waiting line. As an example, consider what we may refer to as a "bank system" and a "supermarket system." In a bank system, there is a single queue and thus a single multi-server system. In contrast, a supermarket features multiple waiting lines, hence we deal with multiple single-server systems.

Let us deal with multiple single-server systems first as they are a straightforward extension of the concepts discussed earlier. The usual assumptions include no balking (i.e., customers in need of service will join the queue regardless of its length) and no jockeying (i.e., customers do not change queues if they perceive that another queue may result in shorter waiting times). As an example, suppose again that arrivals follow a Poisson distribution with a mean arrival rate of $\lambda = 45$ customers per hour, while service time is exponentially distributed with an average service time of $1/\mu = 2$ minutes (or, equivalently, $\mu = 30$ customers per hour). Before performing any computations, feasibility requires that $\rho = \lambda/\mu = 45/30 = 1½$ does not exceed the number of service stations c, requiring at least two service stations in this instance. Suppose now that $c = 3$ service stations are available. As a result, we now deal with three separate single-service systems, or,

in terms of the taxonomy, 3 × *M*/*M*/1 systems. Consider now the customers. While an average of $\lambda = 45$ customers are in need of service, each of the three systems receives only about one third of this number, as customers may be assumed to randomly choose the system they want to be served by. Hence, for each of the three systems, we have an effective arrival rate of $\lambda' = 15$. The value λ' will then replace λ in all of the formulas. In our example, feasibility is guaranteed as $\rho' = \lambda'/\mu = 15/30 = ½ < 1$ for each of the systems. We can then apply the usual steady-state formulas for *M*/*M*/1 models shown in Table 1. For instance, the average time a customer spends in the system is $W_s = \frac{1}{\mu - \lambda'} = 1/(30–15) = 1/15$ hours = 4 minutes. On the other hand, the average number of customers waiting is $L_q = \frac{\rho^2}{1-\rho} = (½)^2/(1–½) = ½$, meaning that on average half a customer is waiting in line in each of the three subsystems. In other words, on average in all of our three service systems combined, 1.5 customers will be waiting for service.

Next, consider a single *M*/*M*/3 system such as the one we encounter in a bank with three tellers. Some of the relevant formulas are summarized in Table 12.2, where the values of the waiting times $W_\bullet$ can be computed by using Little's formula (2).

Table 12.2: Formulas for *M*/*M*/*c* systems

$P_0 = \frac{1}{\left[\sum_{i=0}^{c-1} \frac{\rho^i}{i!}\right] + \frac{\rho^c}{c!(1-\frac{\rho}{c})}}$	$P_n = \begin{cases} \frac{\rho^n}{n!} P_0, \text{if } 0 \le n \le c \\ \left(\frac{\rho^n}{c! c^{n-c}}\right) P_0, \text{if } n > c \end{cases}$
$L_q = \frac{\rho^{c+1}}{(c-1)!(c-\rho)^2} P_0$	$L_s = L_q + \rho$

In our example, first compute the probability that there are no customers in the system. Here, $P_0 = \frac{1}{[1+\frac{3}{2}+\frac{9}{8}]+\frac{(3/2)^3}{3!(1-\frac{3/2}{3})}} = \frac{4}{19}$. This allows us to compute $L_q = 9/38 \cong .2368$, $L_s = 9/38 + 3/2 \cong 1.7368$ and, applying Little's formula (2), we obtain $W_q = L_q/\lambda = .3158$ minutes and $W_s = L_s/\lambda = 2.3158$ minutes.

The differences between the two models arise from the fact that in the former model with separate queues, it is possible that at least one customer is still waiting, while a service station is idle. This is based on the aforementioned assumption that jockeying is prohibited.

A simple extension is a model with self-service. It is easily derived from the standard *M*/*M*/c model by letting the number of service stations *c* tend to infinity.

Such a model will not include any waiting time, so that $L_q = W_q = 0$ and $W_s = 1/\mu$, so that $L_s = \rho$. It can also be shown that in this case, $P_n = \frac{e^{-\rho}}{n!}\rho^n$ for any $n \geq 0$.

For more complex queuing models, it is useful to have either tables or figures at hand that provide the decision maker with a quick idea of what to expect without engaging in complex computations. Entire books with queuing graphs and queuing tables exist. Figure 12.2 shows a typical example of such queuing graphs. This particular graph depicts the situation of a $c \times M/M/1$ graph, where the abscissa shows the utilization rate ρ', while the ordinate is related to the total number of customers in all systems cL_s'. The steep solid line relates to the case of a single facility (i.e., $c = 1$), the broken, broken-and-dotted, broken-and-twice-dotted, and dotted lines are for $c = 2$, 3, 5, and 10 service stations, respectively.

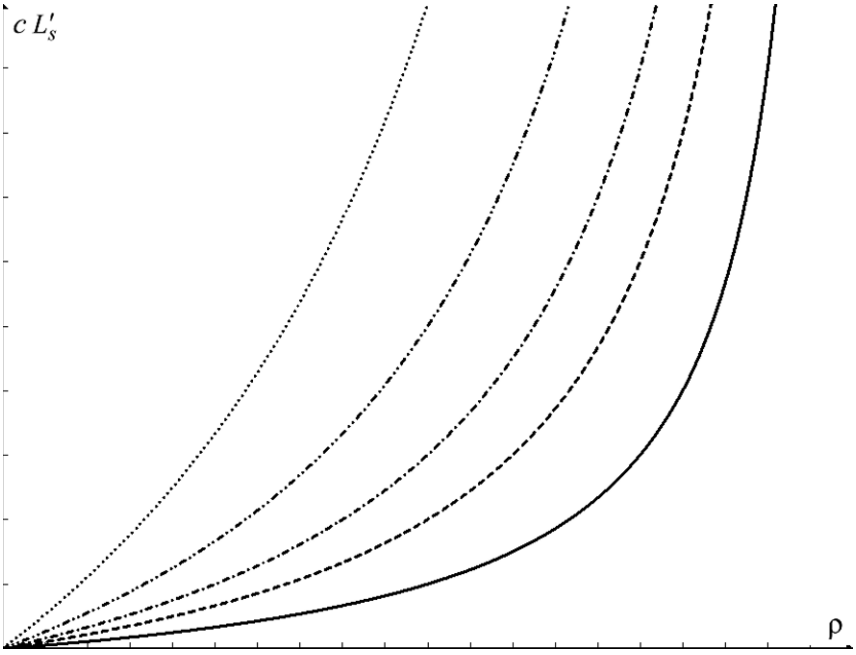


Figure 12.2

Finally, we would like to have a look at multiple-server problems from a slightly different angle. Suppose that the decision maker specifies a certain service level and inquires how many service stations are needed in order to provide a prespecified level of service. Questions of this nature frequently arise in the public service, e.g., how many police officers are needed so as to adequately protect an area. As an illustration, consider the following

Example: An average of twenty customers per hour randomly arrive at a hospital. The system uses a subsystem for each doctor on duty, and there will be an unknown number of c doctors available. Given that each doctor can deal with an average of ten patients per hour, hospital administrators want to ensure that the average waiting time for any patient does not exceed 10 minutes. What is the smallest number of doctors that allows this?

Solution: The system described above is a set of c parallel $M/M/1$ systems, and the effective arrival rate at each of these systems is then $\lambda' = \lambda/c$. Following the results in Table 12.1, the average waiting time is defined as $W_q = \frac{\rho}{\mu - \lambda}$, which, with $\rho' = \frac{\lambda'}{\mu} = \frac{\lambda}{\mu c} = \rho / c$, can be rewritten as $W_q = \frac{\rho / c}{\mu - \lambda / c} = \frac{2/c}{10 - 20/c}$. The condition is now that the average waiting time is no higher than 1/6 [hour], which can be written as

$$\frac{2/c}{10 - 20/c} \leq 1/6.$$

This inequality can be rewritten as $\frac{2}{10c - 20} \leq \frac{1}{6}$, so that $c \geq 3.2$. In other words, at least four doctors are needed to provide the desired service.

It may also be interesting to note that with three doctors, the average waiting time is 12 minutes, while four doctors result in an average waiting time of only 6 minutes.

12.2 Optimization in Queuing

While queuing models are primarily designed to compute performance measures, they can also be applied in the context of optimization. As an example, consider a retail establishment. The owner of the store has to decide how many clerks to employ for the cash registers at the checkout counter. Clearly, increasing the number of clerks will increase the costs. However, at the same time more clerks will result in less waiting time for customers, which, in turn, results in less ill will, lost sales, and other customer behavior detrimental to sales. One of the main problems applying these models is the quantification of the loss due to customer ill will.

The example of a tool crib is much easier to justify. A tool crib is a place in which expensive tools are kept that are not in constant use by the workers. Due to cost considerations, it would not be feasible to provide each worker with one tool, so

that a service desk is established, where workers can sign out the tool whenever it is needed. The costs of the system include the costs of the clerks as well as the costs for the lost time of the workers. If c is the number of clerks and $\$_c$ and $\$_w$ denote the hourly wage of a clerk and a worker, respectively, the costs can then be written as

$$C = (\text{cost of clerks}) + (\text{costs of workers' lost time}) = c\$_c + \$_w L_s.$$

The idea is now to determine the optimal number of clerks so as to minimize the overall costs. As an illustration, consider the following

Example: The demand for a specialized tool occurs randomly at a rate of about 100 times per hour. Whenever the need arises, workers walk over to the tool crib, sign out the tool, use it, and then return it to the tool crib. All clerks are equally efficient with a service time of 3 minutes. For simplicity, we assume that the organization of the signing out follows $c \times M/M/1$ systems. Assume that the hourly wage of a clerk is $\$_c = 10$, while a worker's lost hour costs $\$_w = 25$.

Solution: With the given parameters of $\lambda = 100$ and $\mu = 20$ (and thus $\rho = 5$), we first note that due to the feasibility condition $\rho \leq c$, we will need at least six clerks. The optimal number of clerks can then be determined as follows. Consider the two cost curves that determined the total costs. The costs for the clerks increases linearly with the number of clerks. On the other hand, the cost for the workers' lost time decreases hyperbolically with an increasing number of clerks. This is shown in Figure 12.3. Note that while only integer values of c are relevant, we display the costs for all real values of c so as to better show the shapes of the curves.

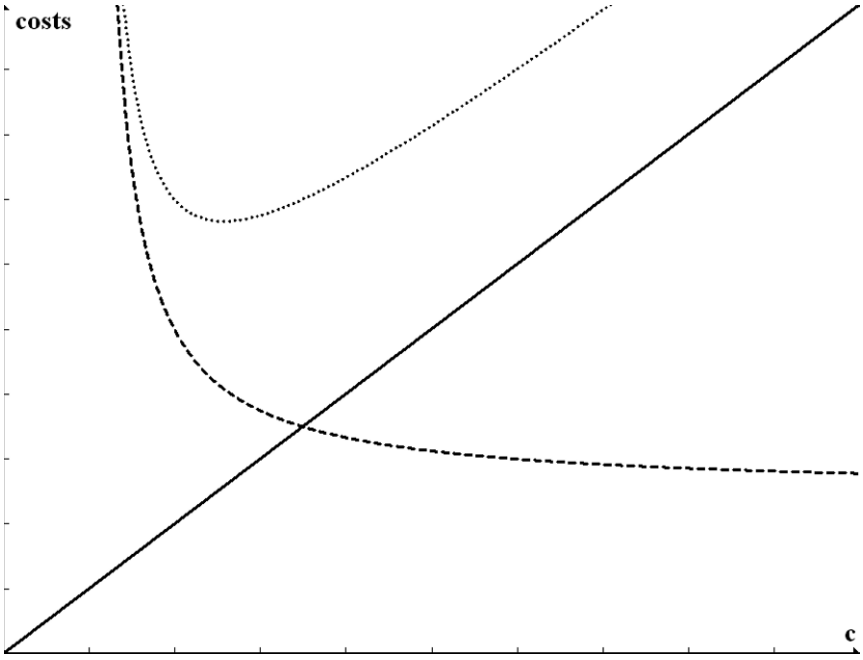


Figure 12.3

For a very small number of clerks (e.g., $c = 6$), waiting time for the workers will be very long, resulting in high costs. As the number of clerks increases, waiting times decrease and with them the costs. The reduction of workers' waiting times due to adding clerks to the system is very significant at first, but less and less so as the number of clerks decreases, Eventually, the benefit of an additional clerk is outweighed by his costs, so that the total costs start increasing again. This suggests a brute force search procedure, in which the total costs are computed for $c = 6, 7, 8, \ldots$ clerks until the costs that were initially decreasing, start increasing again. At that point, the optimal number of clerks has been found. Detailed computations are shown below.

$c = 6$: $\lambda = 100/6 = 16.6667$, $\rho = 16.6667/20 = .8333$, $L_s = 5$ in each system,
Cost = 6(10) + 5(6)(25) = 810.
$c = 7$: $\lambda = 100/7 = 14.2857$, $\rho = .7143$, $L_s = 2.5$ in each system,
Cost = 7(10) + 2.5(7)(25) = 507.5.
$c = 8$: $\lambda = 100/8 = 12.5$, $\rho = .6250$, $L_s = 1.6667$ in each system,
Cost = 8(10) + 1.6667(8)(25) = 413.34.
$c = 9$: $\lambda = 100/9 = 11.1111$, $\rho = .5556$, $L_s = 1.25$ in each system,
Cost = 9(10) + 1.25(9)(25) = 371.25.
$c = 10$: $\lambda = 100/10 = 10$, $\rho = .5$, $L_s = 1$ in each system,
Cost = 10(10) + 1(10)(25) = 350.

$c = 11$: $\lambda = 100/11 = 9.0909$, $\rho = .4545$, $L_s = .8333$ in each system,
Cost = 11(10) + (.8333)7(25) = 339.16.
$c = 12$: $\lambda = 100/12 = 8.3333$, $\rho = .4167$, $L_s = .7143$ in each system,
Cost = 12(10) + (.7143)12(25) = 334.29.
$c = 13$: $\lambda = 100/13 = 7.6923$, $\rho = .3846$, $L_s = .625$ in each system,
Cost = 13(10) + .625(13)(25) = 333.13.
$c = 14$: $\lambda = 100/14 = 7.1429$, $\rho = .3571$, $L_s = .5556$ in each system,
Cost = 14(10) + .5556(14)25 = 334.43.

At this point, the costs start increasing again, so that it is optimal to have $c = 13$ parallel service stations.

Another possibility to incorporate optimization in queuing systems occurs, when retraining of clerks is considered. The basic setting is similar to that of the tool crib above (with the firm paying for service as well as wasted time). The retraining time for the clerks includes the actual (recurrent) retraining as well as costs for the time that the clerk is absent during training, at which time the position must be staffed by other clerks. It is typical that the costs to increase a clerk's service rate increase at an increasing rate. A numerical illustration is provided in the following

Example: Customers arrive at a system at a rate of $\lambda = 30$ per hour. Keeping them waiting is estimated to cost \$20 per hour. At present, the service rate is $\mu = 40$, but with some additional training, this rate can be increased up to 60. Training to achieve a service rate of $\mu \in [40, 60]$ costs $2(\mu - 40)^2$. For simplicity, we assume that an *M*/*M*/1 system is used. To what service rate should the clerk be trained so as to minimize the training and the service costs?

Solution: The cost function under consideration includes again two components. They are the training costs of the clerk and the cost for waiting customers. Following the results in Table 12.1, the average waiting costs in an *M*/*M*/1 system are $L_q = \dfrac{\rho\lambda}{\mu - \lambda} = \dfrac{\lambda^2}{\mu(\mu - \lambda)}$.

Then the cost function is then

$$C = (\text{retraining costs}) + (\text{waiting costs}) = 2(\mu - 40)^2 + 20\ \frac{900}{\mu(\mu - 30)}.$$

The graph in Figure 12.4 shows the total costs in this example. In particular, the function reaches a minimum at $\mu = 41.12358$ with costs $C = 41.8742$.

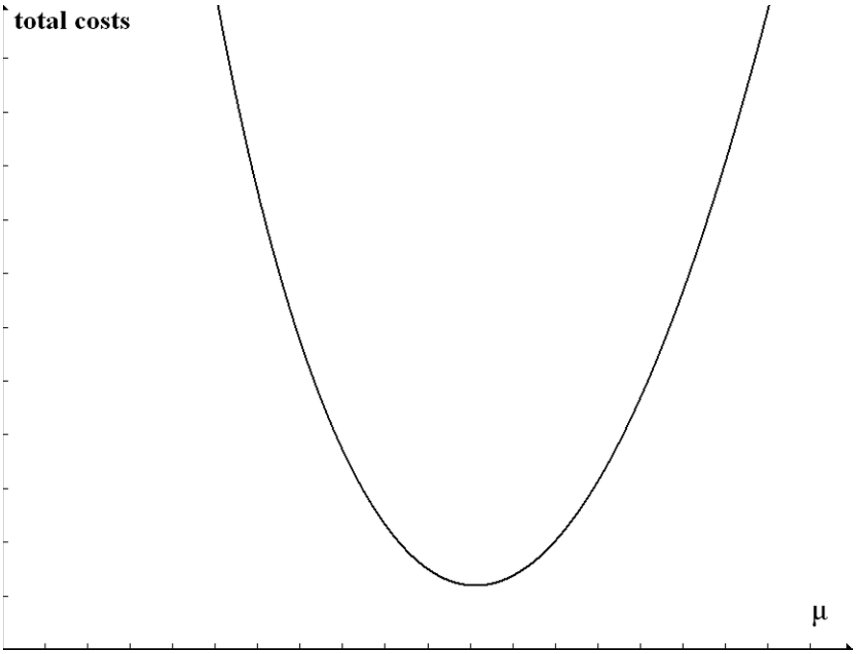


Figure 12.4

Total costs for a number of other service rates have also been computed. They are shown in Table 12.3.

Table 12.3: Queuing costs for differently trained clerks

μ	40	41	42	50	60
C	45	41.9113	43.7143	218	810

In other words, leaving the clerk essentially un(re-)trained will cost $45, less than 10% off optimum, where as training the clerk up to capacity will cost 18 times in total as much as leaving him untrained.

Exercises

Problem 1 (optimization of the number of channels and the service rate): Customers arrive at a retail outlet a rate of 12 per hour. The total time that customers spend in the store contributes to their dissatisfaction. A wasted customer hour has been estimated to cost $20. A clerk at the checkout counter typically earns $8 and can serve up to 10 customers per hour.

(a) What is the cost-minimal number of checkout counters?
(b) Suppose now that an alternative to the system under (a) is to employ two clerks who have been retrained. Their retraining enables them to serve up to 15 customers per hour and they will earn $10 per hour. Is it worth considering this option?

Solution:
(a)
$c = 2$, $\lambda = 6$ each, $\rho = 6/10 = 0.6$, $L_s = 1.5$, so that $TC = 20(2)(1.5)+2(8) = 76$
$c = 3$, $\lambda = 4$ each, $\rho = 4/10 = 0.4$, $L_s = 2/3$, so that $TC = 20(3)(2/3)+3(8) = 64$
$c = 4$, $\lambda = 5$ each, $\rho = 3/10 = 0.3$, $L_s = .4286$, so that $TC = 20(4)(.4286)+4(8) = 66.29$
This implies that the optimal solution is to have 3 clerks. At optimum, the system will cost $64 per hour.
(b) With $\mu = 15$, we obtain $\rho = .4$ and $L_s = \frac{2}{3}$. Then cost $= 20(2)(\frac{2}{3}) + 2(10) =$ $46.67, which is cheaper than the 3-clerk option in (a).

Problem 2 (optimization of the number of channels and sensitivity analysis): Joe plans to open his own gas station "Joe's Place." He has planned to open from 7 a.m. to 11 p.m. He estimates that fifteen customers will arrive each hour during the day to fill up their tanks. Doing so takes typically four minutes plus one minute for paying the bill. Joe now has to decide how many pumps to install. He has read in the industry magazine "Full of Gas" that each hour that a customer waits in line costs $15 in terms of loss of goodwill (i.e., patronizing a different gas station in the future, buying smokes and other emergency items elsewhere, etc.). Also, he has determined that installing a pump costs $100 per day.

(a) Determine the optimal number of pumps Joe should install.
(b) Joe has also heard that there may be a possible gasoline shortage—or at least the perception of one—in the near future. Joe read that in the past, this meant that customers do not really change their driving habits, but fill up their tanks twice as often. Would that change his plans?

Solution: (a) Arrival rate per hour $\lambda = 15$, service time $1/\mu = 4 + 1 = 5$ minutes, or $\mu = 12$ customers per hour. Thus, we need at least $c = 2$ pumps for a steady state to exist.
$c = 2$: $\lambda' = 7.5$ each, $\rho = .625$, $L_q = 1.0417$, so that $TC(c=2) = 2(100) + 16(2)15(1.0417) = 700$,
$c = 3$: $\lambda' = 5$ each, $\rho = .4167$, $L_q = .2976$, so that $TC(c=3) = 3(100) + 16(3)15(.2976) = 514$,
$c = 4$: $\lambda' = 3.75$ each, $\rho = .3125$, $L_q = .1420$, so that $TC(c=4) = 4(100) + 16(4)15(.1420) = 536$,
so that it is optimal to install $c = 3$ pumps.
(b) Arrival rate per hour $\lambda = 30$, service time $1/\mu = 2 + 1 = 3$ minutes (as the fill-up time is now only 2 minutes, since the customers fill up when the tank is half full), or $\mu = 20$ customers per hour. Again, at least $c = 2$ pumps are needed.

$c = 2$: $\lambda' = 15$ each, $\rho = .75$, $L_q = 2.25$,
so that $TC(c = 2) = 2(100) + 16(2)15(2.25) = 1{,}280$,
$c = 3$: $\lambda' = 10$ each, $\rho = .5$, $L_q = .5$,
so that $TC(c = 3) = 3(100) + 16(3)15(.5) = 660$,
$c = 4$: $\lambda' = 7.5$ each, $\rho = .375$, $L_q = .225$,
so that $TC(c = 4) = 4(100) + 16(4)15(.225) = 562$,
$c = 5$: $\lambda' = 6$ each, $\rho = .3$, $L_q = .1286$,
so that $TC(c = 5) = 5(100) + 16(5)15(.1286) = 592.59$.
Under these circumstances, it would be best for Joe to have $c = 4$ pumps. This represents a 9.34% cost increase over the case without the perception of a shortage.

Problem 3 (comparing queuing systems with fast and slow service): Customers arrive at a retail outlet at a rate of 30 customers per hour. The total time that customers spend in the store contributes to their dissatisfaction. A wasted customer hour has been estimated to cost \$10. Management now has two options: either employ one fully trained fast clerk, who is able to serve up to 50 customers per hour, or two less trained slower clerks, who can handle up to 30 customers per hour each. Each of the two clerks would have his own waiting line (the supermarket system). Each of the slow clerks earn \$6 per hour, while the fast clerk is fully aware of his availability, and asks for \$16 per hour.

(a) Should we hire the two slower clerks or the one fast clerk?
(b) A new applicant for the job offers his services. The company tried him out and it turned out that he is able to handle no less than 75 customers per hour. Based on the result under (a), what is the maximal amount that we would we pay him?

Solution: (a) The arrival rate is $\lambda = 30$. The fast clerk offers $\mu = 50$, so that $\rho = 30/50 = 0.6$ and $L_s = \lambda/(\mu - \lambda) = 30/20 = 1.5$. The hourly costs are then (cost for clerk) + (costs for customers) = 16 + 1.5(10) = \$31.
In case of the two clerks, there are two $M/M/1$ systems, each with an effective arrival rate of $\lambda' = 15$. With a service rate of $\mu = 30$ each, we obtain $\rho = 15/30 = 0.5$ each, so that $L_s = 15/(30 - 15) = 1$ each. The hourly costs are then (costs for two clerks) + 2(costs for customers in each system) = 2(6) + 2(1)(10) = 32. As a result, we should hire the fast clerk, even though he charges more than the two slow clerks together and can handle less customer than the two slower clerks combined.
(b) Given a service rate of $\mu = 75$, we obtain $\rho = 0.4$ and $L_s = ⅔$. With an unknown wage w, this results in costs of $w + ⅔(10) = 6⅔ + w$. This amount should not exceed the costs of the best-known solution (a single fast clerk with hourly costs of \$31), so that the bound on the superfast clerk's wage is $6⅔ + w \leq 31$ or $w \leq \$24.33$.

13 Simulation

Simulation is one of the major devices in an operations researcher's toolkit, and there is little doubt that it is among the most flexible and commonly used techniques. In the words of Budnick *et al*. (1988),

> "Simulation is primarily concerned with experimentally predicting the behavior of a real system for the purpose of designing the system or modifying behavior."

In other words, simulation is a tool that builds a model of a real operation that is to be investigated, and then feeds the system with externally generated data. We generally distinguish between *deterministic* and *stochastic simulation*. The difference is that the data that are fed into the system are either deterministic or stochastic. This chapter will deal only with stochastic simulation, which is sometimes also referred to as *Monte Carlo simulation* in reference to the Monte Carlo Casinos and the (hopefully) random outcome of their games of chance.

Another distinction is between *continuous* and *discrete event simulation*. Continuous simulation deals with processes that are continuous and that are modeled as continuous. Typical examples include the growth of plants, movement of vehicles, and temperatures. In contrast, discrete event simulation (the only kind of simulation discussed in this chapter) has a finite number of points of time, during which events occur. This could be the demand for a product during a specific day, the number of times a website is visited, or the number of incidents of a specific disease at a regional hospital.

13.1 Introduction to Simulation

The main reason for a researcher to resort to simulation is twofold. First of all, simulation is probably the most flexible tool imaginable. Take queuing as an example. While it is very difficult to incorporate reneging, jumping queues, and other types of customer behavior in the usual analytical models (see, e.g., Chapter 12 of this volume), this presents no problem for simulation. Similarly, recall that

H.A. Eiselt and C.-L. Sandblom, *Operations Research: A Model-Based Approach*,
DOI 10.1007/978-3-642-10326-1_13, © Springer-Verlag Berlin Heidelberg 2010

the queuing formulas that have been derived refer to steady-state solutions. A system may have to run for a very long time to reach a steady state, assuming that one exists. As a result, a modeler may be more interest in transient states, which are easily available in a simulation.

The second reason is that simulation is very cheap. Building a model that simulates the opening of a new restaurant will most certainly be a lot less expensive than trying it out. Even if costs are no subject, the time frame can be compressed in a simulation. For instance, if we were to observe the demand structure of a product, a long time would be required, so that results would probably be available when the product has become technologically obsolete anyway.

The main steps of a discrete-event simulation include

(1) Building of the model,
(2) Assigning numbers to uncertain events according to their likelihoods,
(3) Generation of uncertain events,
(4) Application of the predetermined policies, and
(5) Evaluation of the results including verification of the model.

The generation of random numbers will be explained in some detail in the next section. The generation of uncertain events and the application of policies to them uses an accounting procedure that is demonstrated on a queuing example and an inventory system in Section 13.3.

Before starting to discuss the generation of random numbers, we would like to discuss their assignment to random events. This is best explained by way of an example. Suppose that the owner of a store has observed the demand for a specific item and has determined that there is a 10% chance that the demand is 20, there is a 30% chance that the demand is 35, a 50% chance that the demand is 50, and a 10% chance that the demand is 60. The task is now to assign random numbers to these random events, so that the likelihood of choosing a random number that is assigned to an event equals the observed probability of the event. In our example, we could use single-digit random numbers. If we generate uniformly distributed random numbers, then all digits are equally likely to come up, i.e., the probability of each digit's appearance is 0.1. We could then assign the digit 3 to the demand of 20, the digits 0, 5, and 8 to a demand of 35, the digits 1, 2, 6, 7, and 9 to a demand of 50, and the digit 2 to the demand of 60. Since each digit has a 10% chance of appearing, randomly generated events will have the different demands come up with the observed probabilities. Alternatively, we could make the assignments of double-digit random numbers (in a somewhat more orderly fashion) as shown in Table 13.1. Again, the numbers assigned to the discrete events reflect the observed probabilities.

Table 13.1: Assignment of random numbers to demands

Demand	20	35	50	60
Probability	.1	.3	.5	.1
Numbers assigned to event	01-10	11-40	41-90	91-00

For the assignment shown in Table 13.1, if the random numbers 15, 27, 81 are drawn, they refer to demands of 35, 35, and 50, respectively. It is apparent that assigning different random numbers to random events—even while preserving their probabilities—will result in different demands being generated. In order to overcome the effects that are due the specific assignment, the process should be repeated very often. For example, if the decision maker is interested in the demands for a product during a 12-month period, we would not generate demands for one year, but for thousands of years, so that differences due to different random number assignments will vanish.

Given the size (more so than the complexity) of the task, it is of little surprise that all simulations are computer-based. While it is possible to write simulations in any all-purpose programming language such as C^{++}, special simulation languages have been around sind the 1960s. Among them are simulation languages such as Simscript, Simula, GPSS (General Purpose System Simulation) and others. There are even specialized simulation languages for specific classes of problems, such as Simfactory.

Whenever a simulation has been performed, the validation of the results is mandatory. Often this means checking the computer code and performing statistical tests. However, the validation of some of the behavioral assumptions and the structure of the model are at least as important. For instance, if it was assumed that customers make a special trip to a gas station, this assumption has to be validated. While this is a task that is typically performed before the simulation takes place, it is sometimes necessary to validate an assumption after the fact. As an example, consider a fast-food chain that attempts to locate a new branch. In addition to behavioral studies before the simulation, it may be very useful to apply the model with all of its assumption to an already existing branch and see whether or not it recreates a known situation. If it does not, the discrepancies will allow the modeler to pinpoint the aspects of the model, in which erroneous assumptions may have been made.

13.2 Random Numbers and their Generation

Random numbers have been around for a long time. Among the earlier systematic efforts to generate random numbers is the work by statistician Tippet, who produces tens of thousands of random numbers derived from the measurements of churches in England. In the mid-1950s the Rand Corporation published a tome “A Million Random Numbers.” Today, we distinguish between *true random numbers*

and *pseudo-random numbers*. Roughly speaking, true random numbers are generated by way of a random process, while pseudo-random numbers are machine-generated by means of a deterministic process. An obvious way to generate true random numbers is to roll dice. Assuming that we have a usual six-sided die which is not loaded or skewed, each side has a chance of 1/6 of coming up, i.e., the probability of each number is 16⅔%. It is not difficult to devise differently-shaped dice that have ten sides, once for each possible digit. Note that for the time being we only deal with uniformly distributed random numbers, i.e., those in which all possible numbers have the same chance of appearing. If two-digit random numbers are sought, use multiple dice, roll them all, and add their numbers. Note, however, that care must be taken: taking, for instance, two standard six-sided dice, rolling them and adding up their numbers, will not result in uniformly distributed results. As an example, the outcome of "2" is only possible, if both dice show a "1," which has a probability of 1/36. On the other hand, an outcome of "8" has a probability of 5/36, as it can be realized from 2 and 6, 3 and 5, 4 and 4, 5 and 3, and 6 and 2.

However, changing the numbers on the faces enables us to use the same process. If the first die has the numbers 0, 1, 2, 3, 4 and 5 on its side, the second has 0, 6, 12, 18, 24, 30, and 36 on its sides, the third has 0, 36, 72, 108, 144, and 180, the fourth die has numbers 0, 216, 432, 648, 864, and 1,080, then the number that results from adding the face values of the four dice is a random number between 1 and 1,295.

Generating random numbers by way of rolling dice may be fun, but it certainly is not a viable method for industrial applications. This is when we resort to machine-generated sequences of random numbers. The basic idea of all of these generators is the same: given a *seed*, i.e., an initial user-determined value, we put this number into a "black box," which uses our input and its internal parameters to generate another number, which is used twofold, as the first random number, and also as the next input into the black box, which uses it to generate the next random number.

Among the first random number generators is the *Midsquare method*, which is said to date back to Nobel Prize laureate John von Neumann. The idea is to start with a seed, square it, retain the middle digits as random number and next input, and continue in this fashion. The main reasoning behind choosing the center part of a number and delete its first last parts is this. The last digit(s) of a number is/are not necessarily random. For instance, if the last digit is 5, the square of the number, regardless what the number is, will have a last digit of 5 as well. Similarly, if the last digit is an even number, then the square of the number will also have an even last digit. As far as the leading digit is concerned, there is much less of a chance of getting an 8 or a 9 than getting a smaller first digit.

As an example of the midsquare method, consider the seed $x_0 = 107364$ and assume that we are interested in five-digit pseudo-random numbers. Squaring this

number results in 115 27028 496, so that our first random number is $x_1 = 27028$, the center of the number shown by the appropriate spacing. Using x_1 as input and squaring it results in 73 05127 84, so that $x_2 = 05127$. Squaring x_2 results in 2 62861 29 and $x_3 = 62861$. The process continues with $x_4 = 51505$, $x_5 = 52765$, and so forth.

The midsquare method is plagued by a multitude of problems, though. Take, for instance, the seed $x_0 = 41$ and generate a sequence of random numbers by deleting the first and last digit after squaring. This results in the sequence $x_1 = 68$, $x_2 = 62$, $x_3 = 84$, $x_4 = 05$, $x_5 = 02$, and $x_6 = 00$, at which time the series has degenerated and will never generate anything but zeroes.

A much better choice are so-called *linear congruence methods*. They work with the function

$$x_i = (a + bx_{i-1}) \bmod c,$$

where a, b, and c are integer parameters, while x_i is the i-th random number as usual. The "mod" function returns the remainder as the result of the division. As an example, consider a number of examples. In case 17 mod 5, we divide 17 by 5, which equals 3 and a remainder of 2, thus $17 \bmod 5 \equiv 2$. Similarly, consider 31 mod 9. Dividing 31 by 9 equals 3 and a remainder of 4, so $31 \bmod 9 \equiv 4$.

Suppose now that we use the parameters $a = 17$, $b = 3$, $c = 101$, and $x_0 = 53$. We can then compute

$$x_1 = [17 + 3(53)] \bmod 101 \equiv 75,$$
$$x_2 = [17 + 3(75)] \bmod 101 \equiv 40,$$
$$x_3 = [17 + 3(40)] \bmod 101 \equiv 36,$$
$$x_4 = [17 + 3(36)] \bmod 101 \equiv 24,$$
$$x_5 = [17 + 3(24)] \bmod 101 \equiv 89,$$

and so forth. It is apparent that the largest number that can be generated in this example will be $c - 1 = 100$. This means that after at most 100 generated numbers, the sequence generated with our parameters will reach a number that has been generated before. And, since the parameters have not changed, the same sequence will be generated over and over again. The number of different random numbers that can be generated before the sequence repeats itself is called the *cycle length* or the *period* of the generator. Typically, the idea is to choose the parameters, so that the period is as long as possible. However, that is not the only criterion for a good set of random numbers. Consider a generator with $a = b = 1$, $c = 10$, and a seed $x_0 = 0$. The generator determines $x_1 = 1$, $x_2 = 2$, $x_3 = 3$, and so forth, until we obtain $x_9 = 9$, and $x_{10} = x_0 = 0$. Thus the cycle length equals 10, but the sequence looks anything but random.

Another obvious criterion a sequence of uniformly distributed random numbers has to satisfy is that each digit will come up about 10% of the time. The above sequence 0, 1, 2, …, 9, 0, … does exactly that. However, the conditional probability of, say, a 5 coming up directly following a 2, is zero, while the probability of a 3 directly following a 2 is 1. This is an obvious test that this particular sequence fails. And that makes it a pseudo-random number. In contrast, remember the roll of a single die. Suppose that a 2 came up on one roll, and a 5 on the next. We keep on rolling, until another 2 comes up. What is then the probability that the next roll will show a 5? With a perfect die, it will be 1/6. With any pseudo-random number, it will be 1, as whenever the same number comes up again, we are in a cycle, which repeats itself.

There is a variety of other tests that random number generators have to pass in order to be reasonable. The parameter c is particularly critical and it is often chosen as a large prime number. Overall, each sequence of numbers generated in this way has a finite period.

Part of the importance of random numbers is not so much that they are used for simulations, but they are also crucial for internet security by way of encryptions, and internet gambling. Given the amount of money involved in these ventures, much is at stake. And the general idea is that random numbers can only truly be generated by a process that involves random elements. Some fairly unusual methods have been included, among others seeds that depend on the number of particles emitted from radioactive elements, atmospheric noise from resistors, and, a patented method, a seed based on random movements observed in lava lamps.

So far, we have discussed random numbers that are uniformly distributed. This is not always desirable. However, fairly simple procedures can be employed to transform uniformly distributed random numbers into random numbers that follow other distributions. The easiest case is to determine random variables that are uniformly distributed on [0, 1[. If numbers with k digits to the right of the decimal point are sought, then each k-digit random number needs to be divided by the largest k-digit number that can be generated, *viz*., $10^k - 1$. Random numbers that are uniformly distributed on [0, 1[are very useful to generate random numbers that follow other distributions.

Consider now Poisson-distributed random numbers. The cumulative density function $F(x)$ of a random variable x that follows a (discrete) Poisson distribution with parameter λ can be found in many works with mathematical tables, e.g., Råde and Westergren (2002). Table 13.2 shows the cumulative functional values for the Poisson distribution with parameter $\lambda = 2.5$.

Table 13.2: Cumulative distribution values for a Poisson distribution with $\lambda = 2.5$

x	0	1	2	3	4	5
$F(x)$	.0821	.2873	.5438	.7576	.8912	.9580

Table 13.2 (continued)

x	6	7	8	9	10
$F(x)$	.9858	.9958	.9989	.9997	.9999

We can now generate Poisson-distributed random numbers by starting with uniformly distributed random numbers. If such a random number falls into an interval $F(x_1)$ and $F(x_2)$, then the Poisson-distributed random number is x_2. As a numerical example, consider the following uniformly distributed random numbers:

.0537 .7406 .5926 .8807 .6603 .7126 .8016 .7973 .9584 .6570 .8457

The first random number is between 0 and $F(x = 0)$, so that a random number of 0 results. The next random number 0.7406 is in the interval [.5438, .7576], so that the next random number equals the x-value of the upper end of the interval, *viz*., 3. Similarly, the next uniformly distributed random number .5926 is also located in the interval [.5438, .7576], so that again the x-value of the upper bound x = 3 results as the next Poisson-distributed random number. Continuing in similar fashion results in the ten random numbers 0, 3, 3, 4, 3, 3, 4, 4, 6, 3, and 4.

Consider now random numbers that follow an exponential distribution with parameter λ. Denoting again uniformly distributed random numbers in [0, 1[by u_i, we can then compute exponentially distributed random numbers x_i by using the formula $x_i = -\ln u_i/\lambda$. Given again the above eleven uniformly distributed random numbers, we can compute exponentially distributed random numbers with parameter λ = 2.5 as

1.1697, 1.2012, .2093, .0508, .1660, .1355, .0885, .0906, .0170, .1680, and .0670.

On the other hand, random variables that follow a standard normal distribution can be generated from uniformly distributed numbers in the [0, 1[interval in pairs. Letting u_i and u_{i+1} denote the i-th pair of uniformly distributed random variables, we can obtain a pair of related standard normally distributed random variables x_i, x_{i+1} by using the relations

$$x_i = \sqrt{-2\ln u_i}\,\sin(2\pi u_{i+1}) \text{ and } x_{i+1} = \sqrt{-2\ln u_i}\,\cos(2\pi u_{i+1})\,.$$

The sequence of standard normally distributed random numbers derived from the first ten uniformly distributed random numbers is

.1962, 2.4104, .0986, 1.0278, .0711, .9083, .0581, .6625, .0298, and .2908.

Random numbers that follow other distributions can be computed as well. In the next section we will demonstrate how these random numbers can be used to model practical situations.

13.3 Examples of Simulations

This section will present two numerical examples of simple simulations. While they focus only on very specific aspects, they should be able to convey some of the fundamental ideas used in real-world simulations.

13.3.1 Simulation of a Waiting Line System

The main goal of simulation is to evaluate existing solutions and policies. In other words, we start with a policy (or a solution), and test how this policy or solution will fare in an uncertain environment. And this uncertain environment is recreated by generating scenarios, given the states of nature and their probabilities that have been observed.

This may best be explained by a few examples. First consider a waiting line system. For simplicity, we will work with a single-channel system. Customers arrive at the service station, so that the interarrival times are uniformly distributed on the integers between 4 and 9. The service times are also uniformly distributed, but on the integers between 5 and 7 (i.e., a service time of 5 minutes has a probability of ⅓, the same as a service time of 6 minutes and one of 7 minutes. The purpose is now to evaluate the performance of the system. The criteria used for that purpose can be manifold. On the customers' side, we could use the probability that a customer will have to wait, the average waiting time, and the average number of customers in the system (really a proxy for the congestion of the system). On the server's side, we could be interested in the average idle time during a workday, and, of course, the cost of the system.

First, we will have to assign events to random numbers. For simplicity, we will use single digits for the interarrival times. A random digit of 4 means an interarrival time of four minutes, a random number of 5 means an interarrival time of five minutes, and so forth. In case a random digit 1, 2, 3, or 0 comes up, we will reject the digit and move on to the next random number.

A similar procedure will be used for service times. Here, the random numbers 5, 6, and 7 denote the actual service times in minutes, all other random numbers will be rejected. A set of uniformly distributed random numbers is shown in Table 13.3. Tables of random numbers can be read from left to right and top to bottom, or right to left bottoms up, or in any other more or less organized way. Here, we scan them row by row from left to right, starting with the first row.

Table 13.3: Random Digits

2049	9135	6601	5112	5266	6728	2188	3846	3734	4017
7087	2825	8667	8831	1617	7239	9622	1622	0409	5822
6187	0189	5748	0380	8820	3606	7316	4297	2160	8973

The scenarios for the first 15 customers are generated in Table 13.4. The first column of the Table 13.4 is the customer number. Column 2 lists the interarrival times that are generated with the random numbers from Table 13.3. The first two digits are 2 and 0, and neither of them are assigned to actual interarrival times (only digits between 4 and 9 are), so that these digits are rejected. The next two digits are 4 and 9. They are assigned to interarrival times of 4 and 9, which are now the interarrival times of the first two customers. The next four digits are 9, 1, 3, and 5. Here, 1 and 3 are unassigned and are rejected, so that only the digits 9 and 5 are usable, they are the interarrival times of the next two customers. This process continues, until we have assigned interarrival times for the first 15 customers. So far we have used random numbers of the first seven of the ten blocks of four digits each in the first row of Table 13.3.

We now use the same process to generate the service times for the first 15 customers. Starting with the second row of random numbers in Table 13.3 (alternatively, we could have simply continued where we left off with the interarrival times and started with the eighth block in the first row), we use only random numbers between 5 and 7 as service times, while rejecting all other digits. In the first block, we keep the first and fourth digit (both 7s), while rejecting the second and third digit (0 and 8), as they are unassigned. In the second block, the first three digits are unassigned, leaving on the fourth digit (with a value of 5). The third block has the first digit unassigned, but the remaining three digits (6, 6, and 7) are usable. The fourth block has all digits unassigned, and so forth. The service times for the first 15 customers are displayed in column 8 of Table 13.4.

We are now ready to perform the simulation. We start with customer 1, meaning the first row of Table 13.4. Given that the system starts operating at, say, 9 a.m., and the interarrival time is 4 minutes, the first customer will arrive at 9:04 (column 3). Since no other customer is presently in the system, service for customer 1 will start immediately (column 7). This means that there was no waiting time (column 5), while there was a 4 minute idle time before this customer's arrival (column 6). Given the service time of 7 minutes (column 8), service on customer 1 will be finished at 9:11 (column 4). For simplicity, we compute the aggregate waiting times (column 9) and the aggregate idle times (column 10) for each customer, so that they are easily available for our evaluations later on.

The computations for customer 2 are similar. Given the interarrival time of 9 minutes, this customer will arrive 9 minutes after customer 1 arrived, meaning at 9:13. Since customer 1 has left the service station at 9:11, there was a 2-minute idle time (column 6), but no waiting time for customer 2 (column 5). As a result, service begins immediately at 9:13 (column 7). Given that the service takes 7 minutes (column 8), customer 2 is done and leaves at 9:20.

This process continues in the same fashion until customer 6 leaves. Consider now customer 7. When the customer arrives at 9:44, customer 6 is still in the system, so that customer 7 will have to wait. More specifically, since customer is finished

at 9:46, customer 7 will wait for 2 minutes (which is recorded in column 5). Service for customer 7 starts at 9:46 (reported in column 7). Also note that so that there is no idle time for the system, which is shown in column 6. Customer 7 will leave the system after 6 minute service time (column 8) at 9:52 (column 4). The process continues in similar fashion for the remaining eight customers.

This is also the point, at which other behavioral aspects could easily be incorporated. Take, for instance, the possibility that a customer reneges, i.e., considers the lineup too long (as a proxy for the expected waiting time) and decides not to wait but go elsewhere instead or return at some other time. While such a behavior would be rather difficult to incorporate in an analytical method, it is easy to do so in a simulation. Here, if customer 7 will only enter the system, if there is no other customer in the system, we would record at this time that one potential customer was not served and continue with the next customer. The number of customers who leave the system unserved is another criterion in the evaluation of the performance of the system.

Table 13.4: Simulation of a queuing system

(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)
1	4	9:04	9:11	0	4	9:04	7	0	4
2	9	9:13	9:20	0	2	9:13	7	0	6
3	9	9:22	9:27	0	2	9:22	5	0	8
4	5	9:27	9:33	0	0	9:27	6	0	8
5	6	9:33	9:39	0	0	9:33	6	0	8
6	6	9:39	9:46	0	0	9:39	7	0	8
7	5	9:44	9:52	2	0	9:46	6	2	8
8	5	9:49	9:59	3	0	9:52	7	5	8
9	6	9:55	10:06	4	0	9:59	7	9	8
10	6	10:01	10:12	5	0	10:06	6	14	8
11	6	10:07	10:18	5	0	10:12	6	19	8
12	7	10:14	10:23	4	0	10:18	5	23	8
13	8	10:22	10:29	1	0	10:23	6	24	8
14	8	10:30	10:37	0	1	10:30	7	24	9
15	8	10:38	10:43	0	1	10:38	5	24	10

Legend:
(1) Customer number
(2) Interarrival time (a random number from the list)
(3) The arrival time of the customer
(4) The time at which service performed for the customer is finished and the customer leaves
(5) The time the customer had to wait (not including service time)
(6) The time that the service station is idle before the customer arrives
(7) Starting time of service for the customer

(8) Service time for the customer (a random number from the list)
(9) The cumulative waiting time, and
(10) The cumulative idle time.

Suppose this very small simulation is to be evaluated. We first observe that there is some idle time in the beginning, while later on the system is very busy. This is hardly surprising: after all, the average interarrival time is 6.5 minutes, while the average service time is 6 minutes, leading to an average traffic intensity of $\rho = .9231$, a clear sign of a very busy system.

We also observe that the total waiting time for all the 15 customers that we have observed (column 9) is 24 minutes. In other words, there is an average waiting time of 24/15 = 1.6 minutes per customer. This information will have to be evaluated by the decision maker, who will have to decide whether or not this time is too long. Note again that the waiting times are not even distributed throughout the day, but get longer later in the day.

Another issue is the idle time of the system. The system has been operating from its opening at 9:00 until the last customer left at 10:43, i.e., for a total of 103 minutes. The total idle time (row 15 of column 10) is ten minutes. This means that the total idle time is $10/103 \approx 10\%$ of the time. Again, this time is clustered during the early stages of the operation.

These, and potentially other, criteria can then be used by the decision maker to evaluate and, if deemed necessary, improve the performance of the system by using appropriate measures.

13.3.2 Simulation of an Inventory System

Another popular area, in which Monte Carlo simulation is applied, deals with inventory management. Again, we first have to observe and quantify all system parameters, and then decide on a policy, meaning assign values to the variables under our jurisdiction. In this example, we distribute a single product, for which there is a daily demand that is uniformly distributed between 0 and 99 units. The lead time is assumed to be 3 days. Note that in more realistic applications, the lead time will also be a random variable. Furthermore, we assume that we have an opening inventory of 200 units. The unit costs are \$100 to place an order and receive the shipment, the daily holding costs are 10¢ per unit that is held overnight, and the shortage costs are 30¢ for each unit that is not available and for which a customer has to wait overnight. Unit holding costs and shortage costs are linear in time, meaning that if a customer has to wait for 5 days to obtain the desired good, the shortage costs will be 5(0.3) = \$1.50. The present inventory policy does not place a new order, if another order is still outstanding, even if the inventory level is below the reorder point. Also, in case of shortages, we assume that units are backordered and there are no lost sales. Other than that, management has formulated the following

Policy 1: Place an order of size $Q = 300$, whenever the inventory level observed at the end of a day falls below the reorder point $R = 150$.

In order to perform the simulation, we first need random numbers. Table 13.5 provides the needed numbers, and we will assign a double-digit random number to a demand of the same magnitude, i.e., a random number 67 will symbolize a demand of 67.

Table 13.5: Random numbers

89	91	72	26	10	83	90	30	76	40

Table 13.6 shows the simulation for ten days. Some details of the computations will be elaborated upon below.

Table 13.6: Simulation for Inventory Policy 1

Day #	Inventory level before opening	Demand	Inventory level after closing	Costs: C_o, C_h, C_s
1	200	89	111 → order	100, 11.10, 0
2	111	91	20	0, 2.00, 0
3	20	72	–52	0, 0, 15.60
4	–52 + 300 = 248	26	222	0, 22.20, 0
5	222	10	212	0, 21.20, 0
6	212	83	129 → order	100, 12.90, 0
7	129	90	39	0, 3.90, 0
8	39	30	9	0, 0.90, 0
9	9 + 300 = 309	76	233	0, 23.30, 0
10	233	40	193	0, 19.30, 0

Before opening on Day 1, our inventory is 200 units as stated in the assumptions. The demand on this day (the first two random digits from the list) is 89, so that by the end of the day, our inventory level has decreased to 111. Comparing this value to our reorder point, we realize that the present inventory level has fallen below the reorder point of $R = 150$, so that we place an order. Given the (deterministic) lead rime of 3 days, the shipment that relates to this order will arrive on Day 4 before we open the store. The costs on that day are: \$100 for placing an order, 111(0.10) = \$11.10 in terms of inventory holding costs (the 111 units to be carried over to Day 2 multiplied by the unit holding costs), and zero shortage costs, as no shortages were encountered.

Day 2 is dealt with similarly. Note that even though by the end of the day, only 20 units remain in stock which is much less than the reorder point, a new order will not be placed according to the policy that prohibits placing a new order if another

order is still outstanding. During Day 3, we encounter a shortage. Similar to carrying costs, shortages are assessed for the number of units we are short by the end of the day (here: 52 units short at 30¢ each for $15.60).

This brings us to the morning of Day 4, at which time a shipment with 300 n ew units comes in. This shipment belongs to the order placed on the evening of Day 1. From this shipment, the demand of all customers whose items were on backorder will be satisfied, before the regular demand takes over.

That way, we simulate the process for ten days. At this point, we can evaluate Policy 1. We note that the ordering costs are $200, the carrying costs are $116.80, and the shortage costs are $15.60, for total inventory costs of $332.40. Other characteristics of the system can also be evaluated, for instance the service level, which may be expressed as the proportion of the demand that can be satisfied immediately rather than having to be backordered. For simplicity, we will concentrate on cost considerations. In the simulation of Policy 1, we note that the ordering costs dominate, while shortage costs are very low. This may lead to a revised policy, in which we place larger orders. In particular, we formulate

Policy 2: Place an order of size $Q = 600$, whenever the inventory level observed at the end of a day falls below the reorder point $R = 150$.

Using the same random numbers (and thus the same demand throughout the ten days), the workings of this policy are shown in Table 13.7. Since the calculations are very similar to those in Table 13.6, we just produce the results without further comments.

Table 13.7: Simulation for Inventory Policy 2

Day #	Inventory level before opening	Demand	Inventory level after closing	Costs: C_o, C_h, C_s
1	200	89	111 → order	100, 11.10, 0
2	111	91	20	0, 2.00, 0
3	20	72	–52	0, 0, 15.60
4	–52 + 600 = 548	26	522	0, 52.20, 0
5	522	10	512	0, 51.20, 0
6	512	83	429	0, 42.90, 0
7	429	90	339	0, 33.90, 0
8	339	30	309	0, 30.90, 0
9	309	76	233	0, 23.30, 0
10	233	40	193	0, 19.30, 0

The individual total costs associated with Policy 2 are $C_o = 100$, $C_h = 266.80$, and $C_s = 15.60$ for total inventory costs of $382.40, a 15% increase over Policy 1. We note that the holding costs are now very high, while shortage costs are still at their

previous, low level. A (hopefully improved) new policy may be defined as having a reorder point of, say, $R = 100$, and an order quantity of $Q = 450$. We leave further experimentation to the reader.

In general, we would like to emphasize that the decision rule that is to be evaluated with simulation does not have to be a fixed rule that is determined once in the beginning and then left unchanged throughout the process. Instead, the rule can include periodic updates. For instance, a diet planner could optimize the food plan for a senior citizens' home and, once one or more of the parameters change, reoptimize and implement the new plan. Similarly, we could periodically update the reorder point and order quantity in an inventory system.

In our example, we could formulate the following policy that features dynamic readjustments of order quantity and reorder point:

Policy 3: (1) Starting with an order quantity of $Q = 300$, readjust the order quantity whenever an order is placed, so that it is 3 times the average daily demand since the last time an order was placed (i.e., 3 times the average daily demand in the last cycle).

(2) Starting with a reorder point of $R = 150$, update the reorder point whenever an order comes in, so that $R := R \pm ½$(shortage/inventory level just before the new order comes in). In other words, if there is a shortage just before the arrival of the new order, the new reorder point equals the previous value of R plus half the shortage. If, on the other hand, there is still some inventory left, half of that amount is subtracted from the previous reorder point to obtain the new value of R.

This somewhat more elaborate policy requires some additional explanations. Again, we will use the same random numbers and thus the same demand as in the previous policies. Initially, we have a reorder point of $R = 150$ and an order quantity of $Q = 300$ as in Policy 1. As in Policy 1, the demand on Day 1 equals 89, so that our inventory level has fallen to 111 by the end of the day. The order quantity is now recalculated as three times the average daily demand since the last order was placed. Since this is the first order that we place, the average is computed for the time between the beginning of the simulation and the end of Day 1. Since only one daily demand has occurred, the order quantity is computed as $Q = 3(89) = 267$. Again, the shipment that relates to this order will arrive in the morning of Day 4.

The computations are the same as for Policy 1 until the morning of Day 4 when the shipment arrives. After deducting the backordered demand, we recomputed the reorder point. Since we had a shortage of 52 units before the shipment arrived, the new reorder point is $R = 150 + ½(52) = 176$.

The process continues again until Day 6. In the evening of Day 6, the inventory level has fallen below the new reorder point of $R = 176$, and we place another

order, which will arrive in the morning of Day 9. The order quantity is now computed on the basis of the average demand since the last order was placed. Here, the previous order was placed on Day 1, and the demand since then was 91 units on Day 2, 72 units on Day 3, 26 units on Day 4, 10 units on Day 5, and 83 units on Day 6 for an average of 282/5 = 56.4 units. According to the policy, the order quantity is three times this amount, i.e., 169.2 units, which we round to the nearest integer, so that $Q = 169$.

From here, the inventory system continues without interruption until Day 9, when the shipment arrives that was ordered at the end of Day 6. After satisfying the demand with the backordered items, we still have 145 units in stock. Since there was a shortage of 24 units just before opening on Day 9, the new reorder point is calculated as the previous reorder point of 176 plus half of the latest shortage for $R = 176 + \frac{1}{2}(24) = 188$. Note that even though the opening inventory on Day 9 is below the reorder point, the policy allows orders to be placed only by the end of the day, which is done here at the end of Day 9. Since the last order on Day 6, the daily demand has been 90, 30, and 76 for an average of 196/3 = 65⅓, so that the new order quantity is computed as $Q = 3(65\frac{1}{3}) = 196$.

Table 13.8: Simulation for Inventory Policy 3

Day #	Inventory level before opening	Demand	Inventory level after closing	Costs: C_o, C_h, C_s
1	200 ($R = 150$)	89	111 → order $Q = 267$	100, 11.10, 0
2	111	91	20	0, 2.00, 0
3	20	72	–52	0, 0, 15.60
4	–52 + 267 = 215 ($R = 176$)	26	189	0, 18.90, 0
5	189	10	179	0, 17.90, 0
6	179	83	96 → order $Q = 169$	100, 9.60, 0
7	96	90	6	0, 0.60, 0
8	6	30	–24	0, 0, 7.20
9	–24 + 169 = 145 ($R = 188$)	76	69 → order $Q = 196$	100, 6.90, 0
10	69	40	29	0, 2.90, 0

In this policy, the total ordering costs are \$300, the carrying costs are \$69.90, and the shortage costs are \$22.80 for a grand total of 392.70. Since these costs are about 18% higher than those for Policy 1, further refinements are needed. However, it is worth pointing out that the time frame of ten days is far too short to make any real recommendations. It was chosen here merely for illustrative purposes.

The above two examples were chosen for this book, as they deal with subject matter that was introduced in earlier chapters, and because they are very intuitive. Even these simple scenarios could be extended in a variety of directions so as to become quite involved. Still the basic ideas remain the same regardless of the complexity of the model.

Exercises

Problem 1 (simulation of a replacement problem): The lighting director of a theater is worried about the maintenance and replacement of five floodlights. They fail according to the number of weeks they have been installed and used. The probability that a bulb still functions after it has been used for t weeks (i.e., it is presently in its (t+1)-st week of use) is denoted by $P(Y|t)$. The numerical values are shown in Table 13.9. In addition, the single-digit random numbers associated with the survival events are shown in the last row of the table.

Table 13.9: Conditional survival probabilities and associated random digits

	$P(Y\|0)$	$P(Y\|1)$	$P(Y\|2)$	$P(Y\|3)$	$P(Y\|\geq 4)$
Probability	0.9	0.7	0.5	0.3	0.2
Random numbers	1 – 9	1 – 7	1 – 5	1 – 3	1 – 2

Whenever a bulb fails, it has to be replaced immediately, as “the show must go on.” Changing a bulb individually is expensive, as scaffolding must be put up. The entire process costs \$350. On the other hand, changing all five bulbs once regardless if they still work or not costs \$800.

The director ponders two replacement policies: either replace the bulbs only when they actually fail, or, alternatively, in addition to failures during the week which have to be attended to immediately, change all bulbs every three weeks regardless if they still work or not. It should be pointed out that even if multiple bulbs fail during the same week, they may do so at different times, so that this case has to be treated and paid for as individual failures.

Solution: Use the following random numbers to generate specific instances of survival and failure:

83638 51597 70322 35984 03933 30948 36142 72865 63348 28024

Table 13.10 and Table 13.11 then display for each light its age in a given week, the random number, and an indication, if a bulb works during any given week (W), or if it fails (F). Consider, for instance, Light 4. In week 1, its age is 0 as the bulb is new. According to Table 13.9, the random digits associated with not failing are 1 – 9, and since the random digit is 3, it will symbolize proper functioning. This

means that in the beginning of week 2, Light 4 is of age 2. The next random digit is 9, which, for a bulb in week 2, means failure. This means that during week 2, bulb 4 will be replaced and its age in week 3 will again be 0. The process continues in this fashion for all bulbs. The result of this replacement policy is that 16 bulb replacements are necessary for a total cost of $5,600.

The simulation for the second policy is shown in Table 13.11. Here, we use the same random digits as before. As an explanation of the numbers in the table, consider Light 1. It works during weeks 1 and 2, but fails during week 3, when it has to be replaced during the week. At the beginning of week 4 (indicated by a "*" in the leftmost column), a group replacement is made, at what time the bulb of Light 1 is replaced again, even though it was just replaced individually during the previous week.

It turns out that this policy requires only 12 individual replacements for a total of $4,200, plus three total replacements for $2,400 for a grand total of $6,600. This is more than the costs of individual replacement alone, making the former strategy preferable.

Table 13.10: Simulation given only individual replacements

	Light 1			Light 2			Light 3			Light 4			Light 5		
Week	age	#	*W/F*	age	#	*W/F*	age	#	*W/F*	age	#	*W/F*	age	#	*W/F*
1	0	8	*W*	0	3	*W*	0	6	*W*	0	3	*W*	0	8	*W*
2	1	5	*W*	1	1	*W*	1	5	*W*	1	9	***F***	1	7	*W*
3	2	7	***F***	2	0	***F***	2	3	*W*	0	2	*W*	2	2	*W*
4	0	3	*W*	0	5	*W*	3	9	***F***	1	8	***F***	3	4	***F***
5	1	0	***F***	1	3	*W*	0	9	*W*	0	3	*W*	0	3	*W*
6	0	3	*W*	2	0	***F***	1	9	***F***	1	4	*W*	1	8	***F***
7	1	3	*W*	0	6	*W*	0	1	*W*	2	4	*W*	0	2	*W*
8	2	7	***F***	1	2	*W*	1	8	***F***	3	6	***F***	1	5	*W*
9	0	6	*W*	2	3	*W*	0	3	*W*	0	4	*W*	2	8	***F***
10	1	2	*W*	3	8	***F***	1	0	***F***	1	2	*W*	0	4	*W*
11	2			0			0			2			1		

Table 13.11: Simulation given individual replacements in addition to triweekly group replacements

	Light 1			Light 2			Light 3			Light 4			Light 5		
Week	age	#	*W/F*	age	#	*W/F*	age	#	*W/F*	age	#	*W/F*	age	#	*W/F*
1	0	8	*W*	0	3	*W*	0	6	*W*	0	3	*W*	0	8	*W*
2	1	5	*W*	1	1	*W*	1	5	*W*	1	9	***F***	1	7	*W*
3	2	7	***F***	2	0	***F***	2	3	*W*	0	2	*W*	2	2	*W*
4*	0	3	*W*	0	5	*W*	0	9	*W*	0	8	*W*	0	4	*W*
5	1	0	***F***	1	3	*W*	1	9	***F***	1	3	*W*	1	3	*W*
6	0	3	*W*	2	0	***F***	0	9	*W*	2	4	*W*	2	8	***F***
7*	0	3	*W*	0	6	*W*	0	1	*W*	0	4	*W*	0	2	*W*
8	1	7	*W*	1	2	*W*	1	8	***F***	2	6	***F***	1	5	*W*
9	2	6	***F***	2	3	*W*	0	3	*W*	1	4	*W*	2	8	***F***
10*	0	2	*W*	0	8	*W*	0	0	***F***	0	2	*W*	0	4	*W*
11	1			1			0			1			1		

Problem 2 (evaluation of investment strategies via simulation of stock prices): The manager of an investment company manages a specific fund. There are $1,000,000 available for investment and three stocks, currently priced at $17, $59, and $103 per share, respectively, are considered for that fund. Planning is made on a weekly basis and any amounts that are not invested will be kept in a short-term money-market account that pays 0.01% per week.

The manager considers three investment strategies. The first strategy would be to keep all of the money in the short-term account. This is the benchmark strategy. The second strategy will invest 50% of the available money at the end of any week that has seen at least a 2% increase in value, and will sell at the end of any week that has seen a decline in stock price, provided that a gain can be realized. Otherwise, the stock is held until a gain can be made. The third strategy is to invest 50% of the available money in a stock whenever its price declines, and it will be sold as soon as a gain can be realized. The manager will not purchase new shares of a stock that is still held.

Stock prices are thought to follow two overlapping trends. On the one hand, their value will be determined by the "state of the economy," (measured by the relative value of the currency, unemployment figures, manufacturers' receipts, and similar factors), and, on the other hand, by stock-specific factors. The change of the state of the economy is denoted by Δ, while the changes of the standings of the three industries are Δ_1, Δ_2, and Δ_3, respectively. The stock prices are then thought to be the sum of Δ and Δ_j for stock j, $j = 1, 2, 3$. The probabilities of these changes are denoted by $P(\Delta)$, $P(\Delta_1)$, $P(\Delta_2)$, and $P(\Delta_3)$. These probabilities have been observed and are displayed in Table 13.12 for changes in the overall economy, Table 13.13 for the specific case of the first industry, and Table 13.14 for the second and third industry. In each table, double-digit random numbers # are listed that are associated with the individual changes Δ and Δ_j.

Table 13.12: Probabilities $P(\Delta)$ and random numbers #

Δ	+2%	+1%	±0	–1%	–2%
$P(\Delta)$	0.05	0.15	0.60	0.15	0.05
#	01-05	06-20	21-80	81-95	96-00

Table 13.13: Probabilities $P(\Delta_1)$ and random numbers #

Δ_1	+3%	+1%	±0	–1%	–3%
$P(\Delta)$	0.03	0.15	0.70	0.10	0.02
#	01-02	03-12	13-82	83-97	98-00

Table 13.14: Probabilities $P(\Delta_2)$ and $P(\Delta_3)$ and random numbers #

Δ_2, Δ_3	+3%	+1%	±0	–1%	–4%
$P(\Delta_2)$, $P\Delta_3)$	0.04	0.12	0.60	0.21	0.03
#	01-04	05-16	17-76	77-97	98-00

In addition, we will use the random numbers shown in Table 13.15. They are read row by row from left to right.

Table 13.15: Random numbers

6959915528	3270108890	4539882224	7576293709
1302727211	7576371930	9295108634	6867423785

The task is to evaluate the different investment strategies of the investment manager.

Solution: We first use the random numbers to simulate the states of the economy Δ, followed by the simulation of the states of the three different industries Δ_1, Δ_2, and Δ_3. This allows us to compute the stock prices. All of these computations are shown in Table 13.16.

Strategy 1: Leaving the entire amount of \$1,000,000 in the short-term account for ten weeks will net us \$1,001,000.45, or a 0.1% gain.

Strategy 2: In week 1, none of the stocks has increased by at least 2%, so that we keep our entire amount in the short-term account. By the end of week 2, we have \$1,000,200.01. Since Stock 2 increases by 3% in week 2, we purchase it with half of the available money, i.e., \$500,100. At the price of \$61.38 a share, we obtain 8,147.6051 shares. The first time we can realize a gain is at the end of week 8, at which point we sell the shares at \$61.99 a share for a total of \$505,070.04. This money is kept in the short-term account for two weeks, resulting in a payoff at the end of week 10 in \$505,171.06. The remaining \$500,100 that were not invested in week 2 will remain in the short-term account, resulting in \$504,114.83 for a total of \$1,009,285.89 or an increase of 0.929%.

Strategy 3: By the end of week 1, Stock 3 has declined in value, which leads the investor to invest half of the available money in that stock. The \$1,000,000 has appreciated due to its investment in the short-term account for one week, so that 1,000,100 are available, half of which (\$500,050) are invested in Stock 3. Each share costs \$101.97, so that 4,903.8933 shares are purchased. Since the shares will never exceed that value again during the ten weeks, they will not be sold.

The remaining \$500,050 are left in the short-term account for two weeks until the end of Week 3, when they have appreciated to \$500,150.02. As Stocks 1 and 2 decreased in value in Week 3, half of the available amount is invested in each. (Note, by the way, that Stock 3 decreased in Week 2, but

since we still hold shares of that stock, we do not invest in it again). The sum of $250,075.01 is invested in Stock 1, which costs $16.66 per share, so that we obtain 15,010.5048 shares. We hold them until the end of Week 8, when their price increases to $16.83, which gives us $252,626.80.

Back to the end of Week 3, when we invested $250,075.01 in Stock 2 at $60.77 a share, so that we obtain 4,115.1063 shares. We sell these shares at the end of week 5 for $61.38 each, resulting in $252,585.22. We hold this money in the short-term account until the end of Week 8, when the investment in Stock 1 is liquidated. By that time, we have $505,212.02. Since none of the stocks declined in Week 8, we hold the amount in the short-term account for a week, resulting in $505,262.54. We are now at the end of Week 9. During Week 9, we observed all stocks declining. As we still hold Stock 3, we cannot invest in it, so that we invest the entire remaining money in Stocks 1 and 2 in equal parts. For the 252,631.27 invested in Stock 1, we obtain 15,163.9418 shares, while for the same amount invested in Stock 2, we obtain 4,116.5271 shares of Stock 2.

Since none of the stock prices increases during Week 10, the account by the end of the planning period consists of 15,163.9418 shares of Stock 1, 4,116.5271 shares of Stock 2, and 4,903.8933 shares of Stock 3. The total value of the portfolio is thus $982,989.70, for a loss of 1.7%.

Comparing the three strategies, it appears that the second investment strategy is best.

Table 13.16: Simulation of stock prices

Week	State of economy		Stock 1				Stock 2				Stock 3			
	#	Δ	#	Δ_1	Total	Price	#	Δ_2	Total	Price	#	Δ_3	Total	Price
0						17.00				59.00				103.00
1	69	±0	45	±0	±0	17.00	13	+1%	+1%	59.59	92	−1%	−1%	101.97
2	59	±0	39	±0	±0	17.00	02	+3%	+3%	61.38	95	−1%	−1%	100.95
3	91	−1%	88	−1%	−2%	16.66	72	±0	−1%	60.77	10	+1%	±0	100.95
4	55	±0	22	±0	±0	16.66	72	±0	±0	60.77	86	−1%	−1%	99.94
5	28	±0	24	±0	±0	16.66	11	+1%	+1%	61.38	34	±0	±0	99.94
6	32	±0	75	±0	±0	16.66	75	±0	±0	61.38	68	±0	±0	99.94
7	70	±0	76	±0	±0	16.66	76	±0	±0	61.38	67	±0	±0	99.94
8	10	+1%	29	±0	+1%	16.83	37	±0	+1%	61.99	42	±0	+1%	100.94
9	88	−1%	37	±0	−1%	16.66	19	±0	−1%	61.37	37	±0	−1%	99.93
10	90	−1%	09	+1%	±0	16.66	30	±0	−1%	60.76	85	−1%	−2%	97.93

Appendix A Heuristic Algorithms

In this book, you will hear about or even directly encounter a number of solution algorithms. All of these algorithms fall into two broad categories: *exact algorithms* (sometimes also somewhat misleadingly referred to as *optimal algorithms*) and *heuristic methods* usually simply called *heuristics*. Exact algorithms have the obvious advantage of providing the best possible solution there is, given the user-defined constraints, whereas heuristics do not. Some heuristics do have error bounds, some actually proven, while others are empirical, i.e., they state that a certain heuristic usually (typically on average) finds solutions that have a certain quality. On the other hand, there is computing speed. Some models are such that it takes an exact algorithm exceedingly long to find the optimal solution. Is this relevant? Well, it depends. If the task at hand is to, say, locate a landfill for millions of dollars, you will not care if it takes a laptop two or three weeks to run, so that it can find a solution that may potentially save hundreds of thousands of dollars. There are limits to this argument, of course: if it takes years or even longer to find a solution, most problems have either solved themselves or have become irrelevant by that time. So, this is not acceptable.

In order to make the case for heuristics, consider the situation of automated guided vehicles (*AGV*s). Suppose you have a number of individual work stations on the shop floor, each of which processes a given piece from a semi-finished product to the finished good. For that purpose, it will be necessary to move pieces, on which work has been finished at one station to the next station. Note that it is not necessarily the case that all goods are running through the same sequence of jobs, not all tasks have to be performed at all work stations, and different levels of customization are possible. The movement of goods may be accomplished by automated guided vehicles that receive a message from a workstation whenever a piece is ready for pickup, and the machine knows where the piece has to go next. However, temporarily there may be more pieces to be transported than the machine can handle, so that it will put the tasks in a list. Whenever it is ready it will work on that list, depending on where it is at any point in time, how far it is to the destination for that particular transportation job, how many work stations will be idle if they have to wait for the next job, and many other considerations. It is apparent that solutions to that problem have to be found in real time, i.e., immediately. This is where heuristic algorithms come in.

Heuristic algorithms typically have two phases. The first phase is the *construction phase*, in which a solution is established. This phase starts with nothing, and by the end of the phase, we have a solution. Phase 1 should be followed by Phase 2, which is an *improvement phase*, in which the method uses simple modifications of the present solution that improve the quality of the solution. The best known heuristics are the *Greedy Method* (a construction method), and the *Swap* (or *pairwise exchange*) method, which is an improvement method.

In order to illustrate the Greedy technique, consider the following example. Suppose that a hiker is lost in the woods. In order to be visible from the air for a rescue mission, he decides to climb as high as possible. Since he has neither a map, nor an idea of where he really is (and it is very foggy, so that visibility is very limited, making his vision myopic), he can only determine the shape of the land in close vicinity. At any point, he can examine the terrain to his North, Northwest, West, Southwest, South, Southeast, East, and Northeast. If there are higher points in any of these directions, he will go into that direction that features the highest nearby point (i.e., the best possible improvement, hence the name "Greedy"). If the points in all eight directions are lower than where he is right now, he will conclude that he is standing on top of a hill and stay there, assuming that he has arrived.

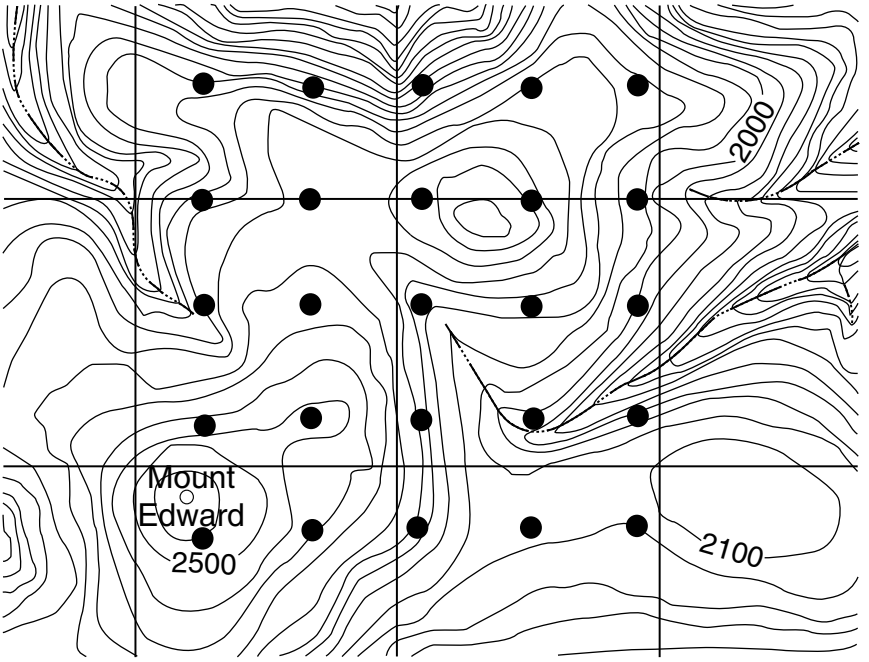
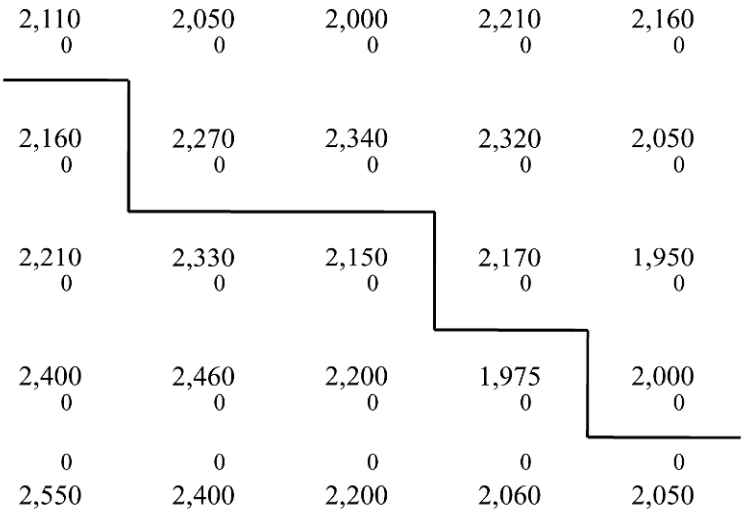


Figure A.1 © Department of Natural Resources Canada. All rights reserved.

As an example, consider the situation shown in Figure A.1. In order to simplify matters, we have determined the altitudes of the points shown as big black dots and have them displayed again in Figure A.2.



Altitudes (in feet)

Figure A.2

Suppose now that the hiker is presently at the Northeasterly point in Figure A.2 at an altitude of 2,160 ft. There are only three surrounding points, those to the West, the Southwest, and the South, with altitudes of 2,210, 2,230, and 2,050, respectively. The highest of these is the point to the Southwest, which is then where the hiker walks to. This point now has eight neighbors, the highest of which is located directly to the West of the hiker's present position at an altitude of 2,340. Consequently, the hiker relocates to this point. Repeating the procedure, the hiker notices that all points in the vicinity of his present location are lower. So, he concludes that he is standing on top of a hill. From the topographic map we know that he is, but we also know that this is not the highest hill in the area of interest.

Formally, a point all of whose neighbors are lower (higher), is called a *local maximum* (*local minimum*). If a local maximum (local minimum) is also the highest (lowest) point overall, it is referred to as a *global maximum* (*global minimum*). As an illustration, consider the function $y = \sin x + 0.05x^2$. For values of x between -10 and $+10$, the function is shown in Figure A.3.

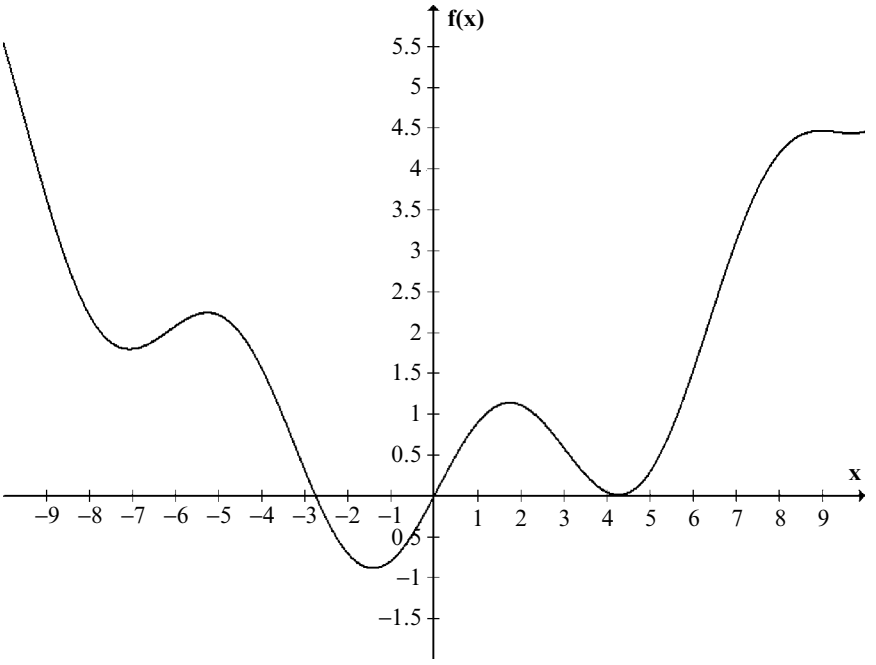


Figure A.3

The function has three minima on the domain shown, they are at $x = -7.0689$ (with $y = 1.7911$), at $x = -1.4755$ (with $y = -.8879$), and at $x = 4.2711$ (with $y = .0079$). (Myopic) heuristics (as well as derivatives) will readily find local optima, but not necessarily global optima. The reason for this is apparent: if any myopic method arrives at, say, the rightmost of the three local minima in Figure A.3, how is the method to know whether or not there are better points located to the left? If a graph such as this were available, it is easy to see, but typically it is not, as almost all problem are multidimensional, and thus cannot be graphed in three or less dimensions.

The hiker's plight described above might now be considered his own problem and of little general interest, if it were not for the fact that each point on the map may represent a course of action (determined by the values of its coordinates), and the contour lines may represent their profit. So the search for the highest hill has now become a search for the point of maximal profit, which is of considerable general interest.

To make the problem even more interesting, had the hiker been in the extreme Southeast and had used the Greedy method for his progress, he would actually ended up at Mount Edward, the overall highest point in the region of interest. As a matter of fact, the line that divides Figure A.2 indicates what is usually referred to as *catchment areas*: if the hiker starts at any point to the Northeast of the line, he

ends up at the local optimum at altitude 2,340 ft, if he starts at any point to the Southwest of the line, he ends up on Mount Edward at the global maximum. A similar situation applies to the function in Figure A.3. Starting a Greedy minimization method any point to the left of $x = -5.2671$ (a local maximum), will end up at the local minimum at $x = -7.0689$. Starting at any point between $x = -5.2671$ and $x = 1.7463$ will end at the global minimum at $x = -1.4276$, and starting anywhere to the right of $x = 1.7463$ will lead to the local minimum at $x = 4.2711$.

This immediately suggests a technique that is called a *multistart method*. The idea is simply to apply a Greedy technique starting at a number of different points, compare the results, and choose the best (highest or lowest, depending on whether a minimum or a maximum is sought).

Next we will discuss an improvement method. The Swap method is easy to describe. Given a solution that has been obtained "somehow," it takes two components and exchanges them. Depending on the specific application, this may mean exchange their sequence, their inclusion/exclusion status, or whatever the problem commands. The method then computes the change of the value of the objective function. If the value has improved (i.e., has increased in case of a maximization problem or decreased in case of a minimization problem), the modified solution becomes the new starting point and the previous solution is discarded. This step is repeated until no further Swap step can improve the solution any further.

As an example, consider the map in Figure A.4 and assume that the task at hand is to determine the shortest route from Memphis, Tennessee, to Reno, Nevada. The traveler has outlined the tour in layers, so that a drive from a city in one layer to a city in the next is about a day's drive.

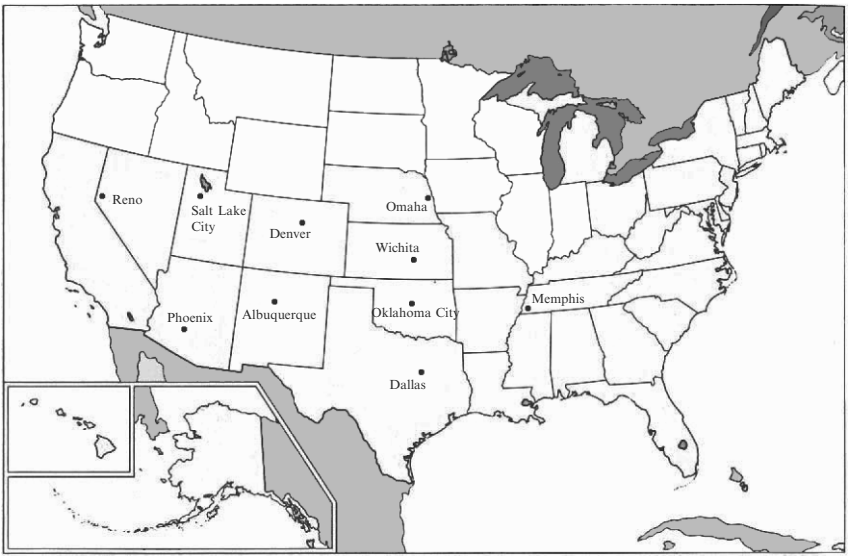


Figure A.4

The distances are as follows: From Memphis to Omaha, Wichita, Oklahoma City, and Dallas we have 650, 555, 460, and 460 miles, respectively. From Omaha to Denver and Albuquerque there are 540 and 860 miles. The distances from Wichita to Denver and Albuquerque are 510 and 585 miles, from Oklahoma City to Denver and Albuquerque are 630 and 550 miles, and from Dallas to Denver and Albuquerque are 780 and 640 miles, respectively. The distances from Denver to Salt Lake City and Phoenix are 505 and 835 miles, while from Albuquerque, there are 610 and 460 miles to Salt Lake City and Phoenix, respectively. Finally, the distance from Salt Lake City to Reno is 525 miles and from Phoenix to Reno there are 735 miles.

In order to obtain some solution, we use the Greedy algorithm, starting at the origin of our trip in Memphis. From here, Greedy will choose the nearest neighbor, which is either Dallas or Oklahoma City, both 460 miles away. We arbitrarily choose Dallas. The nearest neighbor of Dallas is Albuquerque, which is 640 miles away (as opposed to Denver, which is 780 miles from Dallas). From Albuquerque, we take the closest connection to Phoenix (460 miles), and from there we have no choice but take the last long trip to Reno (735 miles). The trip leads us on the route Memphis – Dallas – Albuquerque – Phoenix – Reno, and its total length is 2,295 miles.

At this point we start to swap. One possibility is to swap Phoenix and Salt Lake City. This means that we have to add the connections from Albuquerque to Salt Lake (610 miles) and from Salt Lake to Reno (525 miles), and subtract the connections from Albuquerque to Phoenix (460 miles) and from Phoenix to Reno

(735 miles), for net savings of 60 miles. This is an improvement, and so our new route is Memphis – Dallas – Albuquerque – Salt Lake City – Reno, and its total length is 2,235 miles.

We can now use the new solution and try other Swap moves. For instance, we could attempt to swap Albuquerque and Denver. The net change of such a swap move is +35 miles, so we will not make this change. Another possibility is to swap Dallas and Oklahoma City. The net change is –90 miles, so we make the change and obtain the route Memphis – Oklahoma City – Albuquerque – Salt Lake City – Reno, whose length is 2,145 miles.

At this time we may examine again the pair Albuquerque and Denver. This swap did not improve the solution the last time we tried it, but since then the solution has changed. In fact, swapping the two cities at this point results in a net change of +630 + 505 – 550 – 610 = –25, so that the change is made. this results in the new route Memphis – Oklahoma City – Denver – Salt Lake City – Reno, which is 2,120 miles long.

We may now try to exchange Oklahoma City and Wichita, which leads to a net change of +555 + 510 – 460 – 630 = –25, for another reduction in terms of the total distance, which is now 2,095 miles. The route leads from Memphis – Oklahoma City – Denver – Salt Lake City – Reno. At this point we may try to further reduce the length of the tour, which is no longer possible with swap moves. As a matter of fact (unbeknownst to us when we are just using heuristics), the tour is actually optimal.

Our final example of a heuristic method deals with a much-studied field called *bin packing*. We have an unspecified number of bins, all of which are of the same prespecified length. We also have a number of rods that are to be placed into the bins. The problem is one-dimensional, in that the bins and the rods have the same height and width, so that only the length of the bins and the rods that are placed into them will decide whether or not they actually fit. For instance, if the bin is 20 ft and there are one 6 ft rod, one 3 ft rod, and one 5 ft rod, then these three rods will occupy 14 ft of the bin and leave 6 ft unoccupied. The task at hand is now to put the existing rods into the smallest number of bins possible.

Despite its apparent simplicity, the problem has been proven to be very difficult from a computational point of view. A Greedy-like heuristic is the so-called *First Fit* (*FF*) *Algorithm*. In order to implement the method, we first assume that a sufficiently large number of bins is available. These bins are numbered 1, 2, … . The First Fit Algorithm that can be described as follows:

> *FF Algorithm:* Put the next rod into the bin with the smallest number into which it will fit.

As an example, suppose that all bins are 19 ft long. In addition, we have six 11 ft rods, six 6 ft rods, and twelve 4 ft rods. In this type of situation, we can actually compute a very simple bound for the number of bins that will be needed. Here, the total length of the rods is 6(11) + 6(6) + 12(4) = 150 ft. Given that each bin is 19 ft long, we will need at least 150/19 ≅ 7.89 bins. Since the number of bins must be integer, we will need at least 8 bins. This also means that if we were to find a solution to the problem that requires 8 bins, this solution must be optimal.

Apply now the First Fit Algorithm. Assigning the rods in order of their lengths (i.e., the 11 ft rods first, then the 6 ft rods, and finally the 4 ft rods), we notice that only a single 11 ft rod fits into each bin. This means that we have to put each of the 11 ft rods into one bin each, so that we now have dealt with all 11 ft rods and have used parts of six bins. Next, we assign the six 6 ft rods. Since each of them fits into one of the already partially used bins, we now have six bins with one 11 ft and one 6 ft rod each, leaving 2 ft of free space in each of the six bins. This is not sufficient for any of the remaining 4 ft bins, so that we have to use additional bins. We can place four 4 ft rods in each bin, leaving an empty (and unusable for us) space of 3 ft each, which requires another three bins. We now have assigned all rods to the bins. This solution requires a total of nine bins. There is no apparent pairwise exchange (swap) step able to improve the solution.

On the other hand, if we were to put one 11 ft and two 4 ft rods into each of six bins, this would leave no empty space at all. The remaining six 6 ft rods can be put into two bins. Having again assigned all rods, this solution requires only eight bins and, given the bound computed above, must be optimal.

A variety of other heuristics exists for this problem. An excellent (albeit difficult) pertinent reference is the book by Garey and Johnson (1979). An interesting extension of the problem makes available the different rods over time. This is reminiscent of ready times in machine scheduling. The solution obtained in such a case will be no better than the one found in the case in which all rods are available at the beginning of the process (simply because the problem that makes rods only available over time is more restrictive than the problem discussed here.

The efficiency of the heuristics and their performance may be evaluated in different ways. An obvious evaluation is to use simulation (see Chapter 13 of this book). This will enable users to specify an average or expected *error bound* of the algorithm. For instance, in the above example the heuristic method uses 9 instead of the optimal 8 bins, i.e., 12.5% more than optimal.

However, in some cases it is possible to determine theoretical error bounds, i.e., bounds that cannot be violated. For instance, it has been shown that the solution found by the First Fit Heuristic cannot be worse than about 70% higher than the optimal solution. One problem associated with the theoretical bounds is that while they represent a reliable, provable property, they tend to be very high.

Many other heuristics have been presented in the literature. Many improvement algorithms are *neighborhood searches*, whose main distinguishing feature is that they start with a given solution and search for better solutions in the neighborhood of the present solution. The Swap Method described above belongs to this class. Another very successful heuristic in this class is *tabu search*. The idea of this method is to get out of a local optimum by temporarily allowing the current solution to deteriorate. In order to avoid cycling between solutions that are better and those that are worse, a list of prohibited moves (a tabu list) is set up that is updated as the algorithm progresses. This procedure allows to "get over the hump" from the present point to other solutions that are hopefully better than the best solution known at this point. For example, in Figure A.3, if the best known minimum is $x = -7.0689$, we may allow worse solutions (i.e., those with higher functional values) in our move to the left. This may allow us to find the global minimum at $x = -1.4755$.

Other techniques are based on observations made in the technical or the natural world. Examples are *simulated annealing*, a technique modeled after the way molten metal cools. Similar to tabu search, it allows moves to solutions worse than the best presently known solution. Such moves are allowed with a certain probability that decreases during the course of the algorithm. The formulas ensure that the probability to accept a move to a very bad solution is very small. Other methods follow some behavioral patterns of ant colonies or bees.

Appendix B Vectors and Matrices

Appendices B and C are is intended to provide the reader with some basic refresher regrding some basic operations that involve matrices and vectors and the solution of systems of simultaneous linear equations. It is not designed to replace a text, but as a mere quick reference for some material used in this book.

Definition B1: An $[m \times n]$-dimensional matrix

$$\mathbf{A} = (a_{ij}) = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}$$

is a two-dimensional array of elements a_{ij} arranged in m horizontal rows and n vertical columns, so that the element a_{ij} is positioned in row i and column j. If $m = n$, the matrix is said to be *square*, if $m = 1$, it is called a *row vector*, if $n = 1$, it is a *column vector*, and if $m = n = 1$, it is a *scalar*.

It is common practice to denote scalars by italicized letters, vectors by lower-case bold letters, and matrices by capitalized letters in boldface.

Definition B2: Given an $[m \times n]$-dimensional matrix **A** and an $[n \times p]$-dimensional matrix **B**, the product **C** = **AB** is an $[m \times p]$-dimensional matrix $\mathbf{C} = (c_{ij})$, such that

$$c_{ij} = a_{i1}b_{1j} + a_{i2}b_{2j} + \ldots + a_{in}b_{nj}, \text{ for } i=1, \ldots, m \text{ and } j=1, \ldots, n.$$

Example B1: With **a** = [3, $\sqrt{2}$, –6] and

$$\mathbf{B} = \begin{bmatrix} 0 & 5 & \pi \\ 7 & -6 & 2 \\ \sqrt{3} & -11 & 1 \end{bmatrix},$$

we obtain $\mathbf{aB} = [\, 7\sqrt{2} - 6\sqrt{3},\ \ 81 - 6\sqrt{2},\ \ 3\pi + 2\sqrt{2} - 6\,]$, and

$$\mathbf{B}^2 = \mathbf{BB} = \begin{bmatrix} 35 + \pi\sqrt{3} & -11\pi - 30 & 10 + \pi \\ -42 + 2\sqrt{3} & 49 & -10 + 7\pi \\ -77 + \sqrt{3} & 55 + 5\sqrt{3} & -21 + \pi\sqrt{3} \end{bmatrix}.$$

Definition B3: Given an $[m \times n]$-dimensional matrix $\mathbf{A}$, the *transpose* $\mathbf{A}^T = (a_{ij}^T)$ is the [n × m]-dimensional matrix with $a_{ij}^T = a_{ji}$ for $i = 1, \ldots, m$ and $j = 1, \ldots, n$.

Example B2: Use the vector **a** and the matrix **B** of Example 1, we then obtain

$$\mathbf{a}^T = \begin{bmatrix} 3 \\ \sqrt{2} \\ -6 \end{bmatrix} \text{ and } \mathbf{B}^T = \begin{bmatrix} 0 & 7 & \sqrt{3} \\ 5 & -6 & -11 \\ \pi & 2 & 1 \end{bmatrix}.$$

Appendix C Systems of Simultaneous Linear Equations

Definition C1: A mathematical relation is written as

LHS R RHS,

where *LHS* denotes the left-hand side, *R* is the relation, and *RHS* is the right-hand side of the relation.

Typically (but not necessarily), *LHS* is a function $f(x_1, x_2, \ldots, x_n)$ of n variables $x_1, x_2, \ldots, x_n$, *R* is a relation of type $<, \leq, =, \geq, >$, or $\neq$, and *RHS* is a scalar.

Definition C2: A relation $f(x_1, x_2, \ldots, x_n)\ R\ b$ is said to be *linear*, if the function f can be written as $f(x_1, x_2, \ldots, x_n) = a_1x_1 + a_2x_2 + \ldots + a_nx_n$. We will refer to $f(x_1, x_2, \ldots, x_n)$ as *LHS*, while *b* is the *RHS*.

Example C1: The relation $2x_1 - .7x_2 + \sqrt{11}\, x_3 \leq 59$ is linear, whereas the relations $2\sqrt{x_1} - 0.7x_2 + \sqrt{11}\, x_3 \leq 59$, $2x_1 - 0.7x_1x_2 + \sqrt{11}\, x_3 \leq 59$, and $2x_1 - 0.7x_2 + \sqrt{11}\, x_3^3 \leq 59$ are not, due to the appearance of the square root $\sqrt{x_1}$, the product x_1x_2, and the cubic x_3^3, respectively.

We will first deal with the case when the relation *R* is an equation. Assume now that we have a system of *m* linear equations with *n* unknowns, and that we want to find a solution, i.e., an assignment of values to the variables, that satisfies all equations. It is now possible to prove the following

Theorem C1: Consider a system of *m* simultaneous linear equations in *n* variables:

$$
\begin{aligned}
a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n &= b_1 \\
a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n &= b_2 \\
\vdots \qquad\qquad\qquad\quad & \vdots \quad \vdots \\
a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n &= b_m
\end{aligned}
$$

The system has either no solution, exactly one solution, or an infinite number of solutions.

Example C2: Consider the system

$$
\begin{aligned}
2x_1 + 3x_2 &= 7 \\
4x_1 + 6x_2 &= 10.
\end{aligned}
$$

It is known that if we simultaneously multiply right-hand side and left-hand side of an equation, we do not change its content. Multiplying the first equation by 2, we obtain $4x_1 + 6x_2 = 14$. Now the left-hand side of this equation and that of the second equation are equal, but its right-hand side differs, indicating that there is an inherent contradiction in the system. Thus it is no surprise that the system has no solution.

On the other hand, consider the system

$$
\begin{aligned}
2x_1 + 3x_2 &= 7 \\
4x_1 + 6x_2 &= 14.
\end{aligned}
$$

Multiplying both sides of the first equation by 2 results in the second equation. In other words, the two relations have exactly the same informational content. This means that we really have only a single equation, and since we need one equation to specify the value of each unknown. Hence this system has an infinite number of solutions x_1 and $x_2 = \frac{1}{3}(7-2x_1)$.

Consider now the case that has exactly one solution. Here, we are not concerned with conditions in which a system has exactly one solution, but our focus is on how to actually obtain such a solution, given that it exists. There are many different versions of the *Gaussian elimination technique* (named after the German mathematician Carl Friedrich Gauss, 1777-1855). In order to illustrate the technique, consider the system of simultaneous linear equations

$$
\begin{aligned}
2x_1 + 3x_2 - 5x_3 &= \ \ 1 \qquad (I) \\
\ \ x_1 - 2x_2 + 4x_3 &= -3 \qquad (II) \\
4x_1 + \ \ x_2 + 6x_3 &= \ \ 2 \qquad (III)
\end{aligned}
$$

The idea is to first eliminate one variable, say x_3, from all equations but one. The system has then one equation in all (here: three) variables, whereas two equations include only the remaining variables (here: x_1 and x_2).Among these remaining variables, we now choose another variable to be eliminated (here: x_2), and the

procedure is repeated. In the end, we have a single equation in just one variable, which is then replaced by its value in all of the remaining equations, resulting in another system, in which again one of the equations is just a function of a single variable, which is replaced by its value everywhere, and so forth, until the system is solved.

Applying this idea to our example, we fist eliminate the variable x_3 from equation (*II*) by multiplying (*I*) by 4 and multiplying (*II*) by 5, adding them, and then replacing equation (*II*) by 4×(*I*) + 5×(*II*). The revised system can then be written as

$$\begin{aligned} 2x_1 + 3x_2 - 5x_3 &= 1 && (I) \\ 13x_1 + 2x_2 &= -11 && (II') = 4\times(I) + 5\times(II) \\ 4x_1 + x_2 + 6x_3 &= 2 && (III) \end{aligned}$$

Next, we eliminate x_3 from equation (*III*) and replace equation (*III*) by 6×(*I*) + 5×(*III*). This results in the system

$$\begin{aligned} 2x_1 + 3x_2 - 5x_3 &= 1 && (I) \\ 13x_1 + 2x_2 &= -11 && (II') \\ 32x_1 + 23x_2 &= 16 && (III') = 6\times(I) + 5\times(III) \end{aligned}$$

Since the equations (*II′*) and (*III′*) now contain only the two variables x_1 and x_2, we can eliminate x_2 from (*III′*) by replacing (*III′*) by 23×(*II′*) − 2×(*III′*). This process results in

$$\begin{aligned} 2x_1 + 3x_2 - 5x_3 &= 1 && (I) \\ 13x_1 + 2x_2 &= -11 && (II') \\ 235x_1 &= -285 && (III'') = 23\times(II') - 2\times(III'). \end{aligned}$$

We say that the system is now in triangular form, due to the pattern of coefficients on the left hand side. This allows us to obtain the values of the variables in a recursive procedure. First we can determine the value of x_1 from equation (*III″*). Clearly, $x_1 = -\frac{285}{235} = -\frac{57}{47}$. Inserting the value of x_1 into equation (*II′*) allows us to solve for the variable x_2. In particular, we have $13(-\frac{57}{47}) + 2x_2 = -11$ results in $x_2 = \frac{112}{47}$. Finally, inserting the values of x_1 and x_2 into equation (*I*), we can solve for x_3. The relation reads $2(-\frac{57}{47}) + 3(\frac{112}{47}) - 5x_3 = 1$, which results in $x_3 = \frac{35}{47}$. The system has now been completely solved. The solution is $[x_1, x_2, x_3] = [-\frac{57}{47}, \frac{112}{47}, \frac{35}{47}] \cong [-1.2128, 2.3830, 0.7447]$. Inserting the values of the unknowns into the original equations (*I*), (*II*), and (*III*), we can verify that the solution is indeed correct. In analogous fashion, we can determine the solution of any system of linear equations. For further details, see any pertinent introductory text on linear algebra or the short summaries in Eiselt and Sandblom (2007) or (2004).

Appendix D Probability and Statistics

While most chapters in this book deal with deterministic models, some include probabilistic models, in which concepts of probability are needed. This appendix is intended to briefly cover those probabilistic concepts needed in this book. It is by no means intended to replace a thorough knowledge of statistics.

Definition D1: The *probability* of an event (or outcome) of a random experiment is a number p between 0 and 1, which measures the likelihood that the event will occur.

A value of p close to 0 indicates that the event is unlikely to happen, while a value of p close to 1 means that the event is very likely. We can interpret the probability as the proportion of times that the even or outcome will occur, if the experiment is repeated a large number of times.

Suppose now that the events are *mutually exclusive* (i.e., the events are distinct and do not overlap) and *collectively exhaustive* (meaning that exactly one of them will occur). We then obtain the following result

Theorem D1: If $p_1, p_2, \ldots, p_n$ denote the probabilities of the mutually exclusive and collectively exhaustive outcomes of an experiment, then $p_1 + p_2 + \ldots + p_n = 1$.

Example D1: Consider an experiment that involves tossing a fair coin three times. suppose that each outcome is either head or tail, standing on edge does not occur. Denote H for "head" and T for tail, the outcome "first tail, then head, then tail" is written as THT. Then there are eight possible outcomes: HHH, HHT, HTH, HTT, THH, THT, TTH, and TTT. Given a fair coin, all outcomes are equally likely, and since $p_1 + p_2 + \ldots + p_8 = 1$, we obtain the result that each outcome has a probability of $P(HHH) = P(HHT) = \ldots = P(TTT) = \frac{1}{8}$.

The probability of a *composite event* that consists of several outcomes is the sum of probabilities of the outcome of the event. For instance, the event "obtain exactly one tail in three flips of a fair coin" refers to the event $\{HHT, HTH, THH\}$, so that

the probability of such an event is P({*HHT*, *HTH*, *THH*}) = $\frac{1}{8} + \frac{1}{8} + \frac{1}{8} = \frac{3}{8}$. The *event space* is defined as the set of all possible events of an experiment.

Definition D2: A *random variable X* is a function defined on the event space of an experiment.

Example D2: Given the above coin tossing experiment, let *X* denote the number of heads that come up in the three tosses. There are four possibilities: $X = 0$ (which occurs only in the event {*TTT*}, so that the probability of this event is $P(X=0) = \frac{1}{8}$), $X = 1$ (which happens if one of the events {*HTT*, *THT*, *TTH*} occurs, so that the probability $P(X=1) = \frac{3}{8}$), $X = 2$ (an event that occurs if one of {*HHT*, *HTH*, *THH*} occurs, so that $P(X=2) = \frac{3}{8}$), and finally $X = 3$ (which occurs only if {*HHH*} happens, so that $P(X=3) = \frac{1}{8}$). Again, the sums of all possible events add op to 1.

In general, we distinguish between two different types of random variables, *discrete* and *continuous random variables*.

Definition D3: A random variable *X* is called *discrete*, if it can assume only one of (countably many) values $a_1, a_2, \ldots$. The function $P(a_j) = P(X=a_j) = p(a_j)$ is called the *discrete probability distribution* (function) of *X*. The function $F(a_j) = P(X \le a_j)$ is called the c*umulative probability distribution* (function) of *X*.

Example D3: Again, define the random variable *X* as the number of heads in three tosses of a fair coin. Table D1 shows the probability distribution function for this event.

Table D1: Probability distribution for the coin toss example

X	*P*	*F*
0	$\frac{1}{8}$	$\frac{1}{8}$
1	$\frac{3}{8}$	$\frac{4}{8}$
2	$\frac{3}{8}$	$\frac{7}{8}$
3	$\frac{1}{8}$	$\frac{8}{8}$

We note that $F(a_j)$ is an increasing function of a_j that eventually reaches the value of 1 for the largest value of a_j (in case of finitely many outcomes, and that converges towards 1 for infinitely many outcomes a_j. Since *P* and *F* are probabilities, we must have $0 \le P(a_j) \le 1$ and $0 \le F(a_j) \le 1$ for all events a_j.

Definition D4: A discrete random variable X with a probability distribution function $p(x) = P(X=x) = e^{-\lambda}\frac{\lambda^x}{x!}$, x=0, 1, 2, … is called *Poisson-distributed* with parameter λ. In case a random variable follows this distribution, we will write $X \sim Po(\lambda)$, where *Po* stands for Poisson.

The Poisson distribution is named in honor of Siméon Denis Poisson, a French mathematician, 1781 – 1840. The distribution is of major importance in queuing (Chapter 12 of this book) and will be extensively used in that context.

Example D4: If the random variable $X \sim P_o(\lambda=1.3)$, then $P(X=2) = p(2) = e^{-1.3}\frac{(1.3)^2}{2!} = 0.2303$, and $F(X\leq 1) = p(0) + p(1) = 0.6268$.

Definition D5: A random variable X is called *continuous*, if there exists a function $f(x)$, such that the cumulative distribution function $F(x)$ for X can be written as $F(x) = \int_{-\infty}^{t} f(t)dt$. The function $f(x)$ is called the (*probability*) *density function* of X.

There are two different continuous distributions that are used in this book, the *exponential* and the *normal distribution*.

Definition D6: A continuous random variable X with the density function $f(x) = \lambda e^{-\lambda x}$ with $x \geq 0$ is said to *exponentially distributed* with parameter λ (a positive constant). If X is exponentially distributed, we write $X \sim Ex(\lambda)$.

It is not difficult to demonstrate that the cumulative distribution function of an exponentially distributed variable with parameter λ is $F(x) = 1 - e^{-\lambda x}$.

Definition D7: A continuous random variable X with density function $f(x) = \frac{1}{\sigma\sqrt{2\pi}}e^{-\frac{(x-\mu)^2}{2\sigma^2}}$ is said to follow a *normal distribution* (or, alternatively, *Gaussian distribution*) with parameters μ and $\sigma > 0$. In such a case, we write $X \sim N(\mu, \sigma)$. The distribution $N(0, 1)$, i.e., the normal distribution with $\mu = 0$ and $\sigma = 1$ is called the *standard normal distribution* and is usually denoted by *Z*. Its density function is called the *normal curve* (or *bell curve*), given by the function $f(x) = \frac{1}{\sqrt{2\pi}}e^{-\frac{1}{2}x^2}$.

Definition D8: The *expected value* (or *mean* or *expectation*) $E(X)$ or μ of a discrete random variable X is given by $E(X) = \mu = a_1P(X=a_1) + a_2P(X=a_2) + \ldots$, meaning we multiply each outcome by its associated probability and add them up.

For a continuous random variable X, the expected value is given by $E(X) = \mu = \int_{-\infty}^{\infty} tf(t)dt$.

There are discrete and continuous random variables for which there exists no expected value.

Example D5: For the random variable X of Examples D2 and D3 where X denotes the number of heads in three tosses of a fair coin, we obtain the expected value $E(X) = 0(\frac{1}{8}) + 1(\frac{3}{8}) + 2(\frac{3}{8}) + 3(\frac{1}{8}) = 1\frac{1}{2}$.

Theorem D2: If $X \sim Po(\lambda)$, then $E(X) = \lambda$, if $X \sim Ex(\lambda)$, then $E(X) = \frac{1}{\lambda}$, and if $X \sim N(\mu, \sigma)$, then $E(X) = \mu$.

Definition D9: The *variance* $V(X) = \sigma^2$ of a random variable X with mean μ is defined as $E((X - \mu)^2)$. We call $\sigma = \sqrt{V(X)}$ the *standard deviation* of X.

There are discrete and continuous variables for which the mean exists, but the variance does not. One can show that $V(X) = E(X^2) - \mu^2$ and that $V(X) \geq 0$ for all random variables X, as long as $E(X^2)$, $E(X)$, and $V(X)$ exist.

Example D6: For the random variable X of Examples D.2 and D.3 where X denotes the number of heads in three tosses of a fair coin, we obtain the variance $V(X) = E(X^2) - \mu^2 = 3 - 2.25 = 0.75$.

Theorem D3: If the random variable $X \sim P_o(\lambda)$, then $V(X) = \lambda$, if $X \sim Ex(\lambda)$, then $V(X) = \frac{1}{\lambda^2}$, and if $X \sim N(\mu, \sigma)$, then $V(X) = \sigma^2$.

Theorem D4: If the random variable $X \sim N(\mu, \sigma)$, then $Z = [(X - \mu)/\sigma] \sim N(0, 1)$.

Theorem D5: For any random variable X and for any given numbers $a < b$, we have $P(a < X \leq b) = F(b) - F(a)$. For any continuous random variable X, $P(a \leq X \leq b) = P(a \leq X < b) = P(a < X < b) = F(b) - F(a)$ as well as $P(X = a) = P(X = b) = 0$.

Example D7: Let the random variable $X \sim N(2.3, 0.9)$ and compute the probability $P(1.7 \leq X \leq 2.6)$. We find that $P(1.7 \leq X \leq 2.6) = P\left(\frac{1.7-2.3}{0.9} \leq \frac{X-2.3}{0.9} \leq \frac{2.6-2.3}{0.90}\right) = P(-0.6667 \leq Z \leq 0.3333) = F(0.3333) - F(-0.6667)$. The function $F(x) = \int_{-\infty}^{x} \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}t^2} dt$ is called area under the normal curve and it is tabulated at the end

of this section for values $x \geq 0$. By virtue of symmetry of the normal curve, we find that $F(-x) = 1 - F(x)$, therefore $F(0.3333) - F(-0.6667) = F(0.3333) - 1 + F(0.6667) = 0.6306 + 1 - 0.7475 = 0.3781$ by reading the table and interpolating as necessary.

It has been observed in practice that empirical data that are bell-shaped and symmetric share some common properties. In particular, the *empirical rule* holds, according to which, about 68% of all observations lie within one standard deviation about the mean, about 95% of the observations are within two standard deviations about the mean, and virtually all observations are within three standard deviations about the mean. It is worth mentioning that this is not a provable property, but an observed rule that often occurs in practice.

Making now stronger assumptions about the distribution in particular that it is not only bell-shaped and symmetric, but normal, we are able to confirm the assertion of the empirical rule. In particular, for a standard normal variable $Z \sim N(0, 1)$ we have $F(0) = 0.5$, $F(1) = 0.8413$, $F(2) = 0.9773$, and $F(3) = 0.9987$. Then $P(-1 \leq Z \leq 1) = 2(0.8413 - 0.5) = 0.6826$, $P(-2 \leq Z \leq 2) = 2(0.9773 - 0.5) = 0.9546$, and $P(-3 \leq Z \leq 3) = 2(0.9987 - 0.5) = 0.9974$.

We will now consider events A, B, C, …, which are sets of outcomes of experiments.

Definition D10: Given the events A and B, the *union* $A \cup B$ is the event consisting of outcomes that are in A or in B or both. In contrast, the *intersection* $A \cap B$ is the event that consists of outcomes that are in both A and B.

Theorem D6 (The *addition law for probabilities*): For any events A and B, we have $P(A \cup B) = P(A) + P(B) - P(A \cap B)$.

Example D8: Recall Example D4, in which $X \sim P_o(\lambda=1.3)$. Furthermore, let $A = \{X = 0 \text{ or } 1\}$ and $B = \{1 \text{ or } 2\}$. Then $A \cup B = \{X = 0, 1, \text{or } 2\}$, while $A \cap B = \{X = 1\}$. We then find $P(A) = p(0) + p(1) = 0.6268$, $P(B) = p(1) + p(2) = 0.5846$, $P(A \cup B)$ $p(0) + p(1) + p(2) = 0.8571$, $P(A \cap B) = p(1) = 0.3543$. In accordance with the theorem, we find that $P(A) + P(B) - P(A \cap B) = 0.6268 + 0.5846 - 0.8571 = P(A \cup B)$.

Definition D11: The sets A and B are said to be *collectively exhaustive*, if their union $A \cup B$ includes all possible outcomes of an experiment. The two sets are called *mutually exclusive*, if their intersection is empty. The *complement* $\overline{A}$ of a set A (sometimes also written as $\neg A$) is the set of all possible outcomes not in A.

As far as probabilities are concerned, $P(\overline{A}) = 1 - P(A)$.

The addition low for probabilities can be generalized. For instance, for mutually exclusive sets $A_1, A_2, \ldots A_m$, we obtain $P(A_1 \cup A_2 \cup \ldots \cup A_m) = P(A_1) + P(A_2) + \ldots + P(A_m)$. To establish a multiplication law for probabilities, we need the following

Definition D12: For any events A and B with $P(B) \neq 0$, the *conditional probability of A given B* is

$$P(A|B) = \frac{P(A \cap B)}{P(B)}.$$

Example D9: Consider an experiment, in which the random variable X denotes that at most two heads come up in three tosses of a fair coin, i.e., $X \leq 2$, and define B as an event that sees at least one head in three tosses of a fair coin, i.e. $X \geq 1$. Then $A \cap B = \{X = 1 \text{ or } 2\}$, so that $P(A \cap B) = \frac{3}{8} + \frac{3}{8} = \frac{3}{4}$ and $P(B) = \frac{3}{8} + \frac{3}{8} + \frac{1}{4} = \frac{7}{8}$. Therefore, $P(A|B) = \frac{3/4}{7/8} = \frac{6}{7} \approx 0.8571$.

Theorem D7 (The *multiplication law for probabilities*): For any events A and B, $P(A \cap B) = P(A|B)\, P(B)$.

Note that if $P(B) = 0$, then $P(A|B)$ is not defined, but in this case the right-hand side is interpreted as being zero.

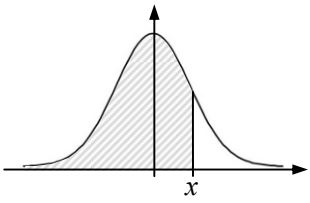
Definition D13: The events A and B are said to be *statistically independent*, if $P(A \cap B) = P(A)\, P(B)$.

Theorem D8 (*Bayes's theorem*): Let the events $A_1, A_2, \ldots A_m$ be mutually exclusive and collectively exhaustive. Then for any event B with $P(B) \neq 0$, we have

$$P(A_i \mid B) = \frac{P(B \mid A_i)P(A_i)}{P(B \mid A_1)P(A_1) + P(B \mid A_2)P(A_2) + \ldots + P(B \mid A_m)P(A_m)} \quad , i=1,\ldots, m$$

The theorem of Bayes (Thomas Bayes, English clergyman, 1702-1761) will be used in Chapter 9 of this book. In the context of Bayes's theorem, the unconditional probabilities $P(A_i)$ are called *prior probabilities*, while the conditional probabilities $P(A_i|B)$ are referred to as *posterior probabilities*.

Area under the normal curve



x	0.00	0.01	0.02	0.03	0.04	0.05	0.06	0.07	0.08	0.09
.00	.5000	.5040	.5080	.5120	.5160	.5199	.5239	.5279	.5319	.5359
.10	.5398	.5438	.5478	.5517	.5557	.5596	.5636	.5675	.5714	.5754
.20	.5793	.5832	.5871	.5910	.5948	.5987	.6026	.6064	.6103	.6141
.30	.6179	.6217	.6255	.6293	.6331	.6368	.6406	.6443	.6480	.6517
.40	.6554	.6591	.6628	.6664	.6700	.6736	.6772	.6808	.6844	.6879
.50	.6915	.6950	.6985	.7019	.7054	.7088	.7123	.7157	.7190	.7224
.60	.7258	.7291	.7324	.7357	.7389	.7422	.7454	.7486	.7518	.7549
.70	.7580	.7612	.7642	.7673	.7704	.7734	.7764	.7794	.7823	.7852
.80	.7881	.7910	.7939	.7967	.7996	.8023	.8051	.8079	.8106	.8133
.90	.8159	.8186	.8212	.8238	.8264	.8289	.8315	.8340	.8365	.8389
1.0	.8413	.8438	.8461	.8485	.8508	.8531	.8554	.8577	.8599	.8621
1.1	.8643	.8665	.8686	.8708	.8729	.8749	.8770	.8790	.8810	.8830
1.2	.8849	.8869	.8888	.8907	.8925	.8944	.8962	.8980	.8997	.9015
1.3	.9032	.9049	.9066	.9082	.9099	.9115	.9131	.9147	.9162	.9177
1.4	.9192	.9207	.9222	.9236	.9251	.9265	.9279	.9292	.9306	.9319
1.5	.9332	.9345	.9357	.9370	.9382	.9394	.9406	.9418	.9430	.9441
1.6	.9452	.9463	.9474	.9485	.9495	.9505	.9515	.9525	.9535	.9545
1.7	.9554	.9564	.9573	.9582	.9591	.9599	.9608	.9616	.9625	.9633
1.8	.9641	.9649	.9656	.9664	.9671	.9678	.9686	.9693	.9700	.9706
1.9	.9713	.9719	.9726	.9732	.9738	.9744	.9750	.9756	.9762	.9767
2.0	.9773	.9778	.9783	.9788	.9793	.9798	.9803	.9808	.9812	.9817
2.1	.9821	.9826	.9830	.9834	.9838	.9842	.9846	.9850	.9854	.9857
2.2	.9861	.9865	.9868	.9871	.9875	.9878	.9881	.9884	.9887	.9890
2.3	.9893	.9896	.9898	.9901	.9904	.9906	.9909	.9911	.9913	.9916
2.4	.9918	.9920	.9922	.9925	.9927	.9929	.9931	.9932	.9934	.9936
2.5	.9938	.9940	.9941	.9943	.9945	.9946	.9948	.9949	.9951	.9952
2.6	.9953	.9955	.9956	.9957	.9959	.9960	.9961	.9962	.9963	.9964
2.7	.9965	.9966	.9967	.9968	.9969	.9970	.9971	.9972	.9973	.9974
2.8	.9974	.9975	.9976	.9977	.9977	.9978	.9978	.9980	.9980	.9981
2.9	.9981	.9982	.9983	.9983	.9984	.9984	.9985	.9985	.9986	.9986
3.0	.9987	.9987	.9987	.9988	.9988	.9989	.9989	.9989	.9990	.9990
3.5	.999767									
4.0	.9999683									
4.5	.99999660									
5.0	.999999713									

References

Axsäter S (2006) Inventory control (2nd ed.). Springer-Verlag, Berlin-Heidelberg-New York

Blumenfeld D (2001) Operations research calculations handbook. CRC Press, Boca Raton, FL

Budnick FS, Mojena R, Vollmann TE (1988) Principles of operations research for management. (2nd ed.) Richard D. Irwin, Inc. Homewood, IL

Chan Y (2005) Location, transport and land-use: modelling spatial-temporal information. Springer-Verlag, Berlin-Heidelberg-New York

Chhajed D, Lowe TJ (2008) (eds.) Building intuition: insights from basic operations management models and principles. Springer Science + Business Media LLC, New York, NY

Dantzig GB (1963) Linear programming and extensions. Princeton University Press, Princeton, NJ

Dantzig GB, Thapa MN (1997) Linear programming: introduction. Springer, New York, NY

Dantzig GB, Thapa MN (2004) Linear programming: theory and extensions. Springer, New York, NY

Daskin MS (1995) Network and discrete location: models, algorithms, and applications. J Wiley & Sons, Inc., New York, NY

Eiselt HA, Sandblom C-L (2000) Integer programming and network models. Springer-Verlag, Berlin-Heidelberg-New York

Eiselt HA, Sandblom C-L (2004) Decision analysis, location models, and Scheduling Problems. Springer-Verlag, Berlin-Heidelberg-New York

Eiselt HA, Sandblom C-L (2007) Linear programming and its applications. Springer-Verlag, Berlin-Heidelberg-New York

Erlenkotter D (1990) Ford Whitman Harris and the economic order quantity model. Operations Research 38: 937–946

Garner Garille S, Gass SI (1981) Stigler's diet problem revisited. Operations Research 49: 1–13

Garey MR, Johnson DS (1979) Computers and intractability: a guide to the theory of NP-completeness. WH Freeman & Co., San Francisco, CA

Gass SI, Assad AA (2005) An annotated timeline of operations research: an informal history. Kluwer Academic, New York, NY

Hillier FS, Lieberman GJ (2010) Introduction to operations research (9th ed), McGraw-Hill/Irwin, New York, NY

Hillier FS, Hillier MS, Schmedders K, Stephens M (2008) Introduction to management science: a modeling and case study approach with spreadsheets (3rd ed.). McGraw-Hill/Irwin, Boston, MA

Lind D, Marchal W, Mason R, Gupta S, Kabadi S, Singh J (2004) Statistical techniques in business and economics. First Canadian Edition, McGraw-Hill, Toronto

Olson DL (1996) Decision aids for selection problems. Springer Series in Operations Research, New York, NY

Råde L, Westergren B (1989) Beta: Mathematics handbook. (2nd ed.) CRC Press, Boca Raton, FL

Walker RC (1999) Introduction to Mathematical Programming. Prentice Hall, Upper Saddle River, NJ

Winston WL (2004) Operations research: applications and algorithms. (4th ed.) Duxbury, Belmont, CA

Website that accompanies this book (includes additional material and problems): http://www.springer.com/business/operations+research/book/978-3-642-10325-4

SUBJECT INDEX

A

ABC classification, 339–340
Active node, 156
Activity
  critical, 263
  duration, 258
Activity-on-arc representation. *See AOA* representation
Activity-on-node representation. *See AON* representation
AILP, 136
Algorithms, exact and heuristic, 417
All-integer linear programming problem. *See AILP*
Allocation problem, 28–32
Anticipated payoffs, 311
AOA representation, 259
AON representation, 259–260
Arc, 177
Arrival rate, 382
Arrival time, 288
Aspiration level, 125
Assignment problems, 48–50

B

Backorders, 349–351
Backward pass, sweep, or recursion, 261–263
Basis point, 89
Bayes's rule, 313
Bayes's theorem, 322–323
Bin packing, 423
Bisection search, 234–235
Blending problems, 39–43
Bottleneck, 94
Branch-and-bound methods, 155–162
Breadth-first-search, 182
Break-even analysis, 8
Breakthrough, 181
Budget constraint, 29

C

Calling population, 380
Catchment areas, 420–421
Center-of-gravity, 237
Center problems, 230–235
Certainty equivalent, 327
Changes
  objective function coefficients, 82–87
  right-hand side values, 87–92
  structural and parameter, 79
Channel, 380
Chinese postman problem, 201
Column
  essential, 222
  dominated, 222
Competitive location, 246
Complementary slackness conditions, 110
Conservation equations, 181
Constraint method, 122–124
Constraints, 5, 16
  addition and deletion of, 81
  binding, 63
  conditional, 146
  tight, 63
Contour lines, 65
Convex hull, 154
Convexity, 69
Corner point, 63
  theorem, 67
Cost, holding or carrying, 342
  ordering, 342
  shortage, 342
Covering matrix, 221
Covering problems, 220–230
CPM, 258–266, 268–272, 277
  project acceleration, 266–272
Crashing, 267–272
Crash time, 267–272
Critical path, 266
Critical path method. *See CPM*

Cut, minimal, 186
Cutting plane method, 154–155
Cutting stock problems, 142–146
Cycle, 178
Cycle length, 343, 399

D

Decision analysis, 305–328
Decision trees, 309–310, 321–324
Decomposition principle, 7
Degeneracy
  dual, 74
  primal, 76
Density function, 435
Depth-first search, 182
Destination, 43
Diet problem, 20–28, 146–148
Dijkstra method, 192–195
Discrete event simulation, 395
Distribution
  exponential, 435
  Gaussian, 435
  normal, 435
  Poisson, 435
Divisibility, 14–15
Duality, 105–112
Dual problem
  setting up, 106–107
  relations to primal problem, 107–112
Due time, 288
Dynamic programming, 3

E

Earliest due date algorithm. *See EDD* algorithm
Earliest possible finishing times. *See EF*
Earliest possible starting times. *See ES*
Economic order quantity. *See EOQ*
EDD algorithm, 292
Edge, 177
EF, 260–261
Efficiency, 325
Efficient point, 120
Either-or constraints, 146
Empirical rule, 437
Employee scheduling, 32–35
EMV, 313–319
EOQ, 343–346
EPII, 325
EPPI, 320
"Equity" objectives, 245
ES, 260–261
Euler, 177, 200
Euclidean distance, 218
Event, 433
EVPI, 319–320
EVSI, 325
Excess variable, 17
Expected monetary value. *See EMV*
Expected payoff with imperfect information. *See EPII*
Expected payoff with perfect information. *See EPPI*
Expected value, definition, 435
Expected value of imperfect (or sample) information. *See EVSI*
Expected value of perfect information. *See EVPI*
Extreme points, 63
  number of, 69

F

Facilities, extensive, 247
  undesirable, 244–245
Feasible direction methods, 69
Feasible
  set, 62
  solution, nonexistence, of 70–72
Feasibility, 7
First fit algorithm, 423–424
Fixed charges, 150–151
Flow, pattern and value, 179
  balancing equations, 181
Flow shop, 288, 298
Flow time, 288
  mean, 289
Floyd-Warshall method, 195–198
Forward pass, sweep, or recursion, 260

G

Games against nature, 306
Gantt chart, 272–275, 291
Gaussian elimination, 430–431
Gini index, 245–246
Goal programming, 124–128
Graph, directed, undirected, and mixed, 177
  connected, 178
Graphical solution method, 60–70
Greedy method, 162–163, 204–205, 228–229, 241–242, 418
Guillotine cuts, 146

H

Halfplane, 61
Halfspace, 62
Heuristic methods, 162–165, 417–425
Hub location, 246
100% rule, 86–87, 92
Hyperplane, 61

I

Implementability, 7
Improvement cone, 118
Incremental technique, 69
Influence diagrams, 307–309
INFORMS, 1
Integer programming, 135–165
Integrality gap, absolute and relative, 140
Interarrival time, 382
Interchange heuristic, 163–165
Interval constraints, 72
Inventory models, 339–363
Inventory position, 341
Investment allocation, 28
Iso-profit lines, 65, 117–118

J

Jackson's rule, 292
Jackson's job shop algorithm, 299–300
Job shop, 288, 299
Johnson's algorithm, 298–299
Just-in-time inventory system, 340

K

Kendall's notation, 380–381
Kirchhoff node equations, 181
Knapsack problems, 140–142
Königsberg bridge problem, 2
Kruskal's method, 199–200

L

Labeling methods, 181ff, 192ff
Land use problem, 148–149
LAPT algorithm, 297–298
Lateness, 289
Layout problems, 247
Lead time, 340, 346–349
  stochastic, 357–362
Linear congruence methods, 399
Linearity, 15
Linear programming, 13ff
  relaxation, 139
List scheduling methods, 294
Little's formula, 383
Location-allocation heuristic, 242–244
Location models, 217–247
Location
  continuous, 217
  discrete, 218
Location set covering problem. *See LSCP*
Logical variables, 146
Longest alternate processing time algorithm. *See LAPT* algorithm
Longest processing time first algorithm. *See LPT* algorithm
Lorenz curve, 245
LPT algorithm, 294–295
LSCP, 221–227

M

Machine scheduling, 287–300
  single, parallel, dedicated, 288
Makespan, 289
Management science, 1
Manhattan distance, 218
Markov chains, 367–376
Markovian property, 368
Marriage problem, 49
Material requirements planning, 340
Matrix, 427–428
Maximal covering location problem. *See MCLP*
Maximal flow problem, 179–186
Maximin rule, 311
MCDM, 115, 306
MCLP, 227–230
McNaughton's algorithm, 295–296
Median problems, 235–244
Midsquare method, 398
MILP, 136
Min cost feasible flow problem, 186
Min-cut max flow theorem, 186
Minimax problems, 152
Mixed integer linear programming problem. *See MILP*
Model, deterministic and stochastic, 14
Modeling process, 9ff
MODI method, 46
Multiattribute decision making, 115
Multicriteria decision making problems. *See MCDM*
Multiobjective programming, 115–128
Multistart procedure, 205, 421

N

Nash equilibria, 246
Neighborhood search, 425
Network, 177
  flows, 179–189
Newspaper boy problem, 313
Nodes, 177
  decision, 324
  event, 324
  reachable, 178
Nonbreakthrough, 181
Nondominated frontier, 121
Nondominated point, 120
Noninferior point, 120
Normal duration, 266

O

Objective, 4
  function, gradient of, 65
Open shop, 288, 297
Operations research
  definition, 1
  elements, 4
  journals, 3
Opportunity costs, 93
Optimality, 7
Optimal solution
  alternative, 74
  economic analysis of, 92–100
  unbounded, 73–74
  unique, 68
Optimist's rule, 312
Origin, 43
Overshipments, 47

P

Parameters, 6
Pareto-optimal point, 120
Path, 178
Payoff table, 306
PERT, 275–280
Pessimistic rule, 311
Phase, construction and improvement, 418
Pollaczek-Khintchine formula, 384
Polytope, 63
Postoptimality analyses, 78–100
Precedence relations, 258
Preemption, 290
Probabilistic, 14
Probability
  conditional, 438
  definition, 433
  distribution, 434
  posterior, 322
  prior, 322
Process, stationary, 369
Processing time, 288
Production–inventory models, 35–39
Production-lot size model, 355–357
Production planning, 18–20
Program evaluation and review technique. *See PERT*
Project networks, 257–280

Q

Quantity discounts, 352–355

R

Random numbers, true and pseudo, 397–398
Random variables, 434
Ready time, 288
Rectilinear distance, 218
Reduced costs, 93
Redundancy, 75–76
Regret criterion, 312
Release time, 288
Reorder point, 347–349
Reshipments, 47
Resource requirement graph, 273–274
Resources, scarce, 28
Review systems, continuous and periodic, 362
Risk, 28ff, 307, 313–315
  aversion, neutral, seeking, 327
Robust optimization, 312
Routing, arc and node, 200
Row, dominated, 222

S

Satisficing, 125
Scalar, 427
Schedule length, 289
Selection problem, 305
Sensitivity analyses, 7, 78, 316–319
  graphical, 78–92
Service
  rate, 382
  station, 380
  time, 382
Shadow prices, 93, 94
Shop, open, flow, job, 288
Shortest path problems, 189–198
Shortest processing time algorithm. *See SPT* algorithm
Simplex method, 69
Simulated annealing, 425
Simulation, deterministic and stochastic, 395
Simultaneous linear equations, 429–431
Sink, 179, 259
Smith's ratio rule, 291
Solution
  definition, 7
  efficient, 120
  nondominated, 120
  noninferior, 120
  pareto-optimal, 120
Source, 179, 259
Spanning tree, 198–200
  minimal, 199–200
SPT algorithm, 290

Standard deviation, 436
State
  space, 367
  absorbing, 368
Statistical independence, 438
Steady state solution, 372, 382
Stochastic processes, 367–376
Subtour elimination constraints, 203
Supply chain management, 340
Swap heuristic, 163–165, 229–230, 418

T

Tabu search, 425
Tardiness, 289
Target value, 125, 314–315
Tool crib, 388–389
Tradeoffs, 115
Traffic intensity, 382
Transient states, 382
Transition probabilities, 368
Transportation problems, 43–48
Transportation problem, (un-)balanced, 44
Traveling salesman problem, 201–205
Tree (graph), 178

U

Uncertainty, 307, 310–313
Undesirable facilities, 244–245
Utility theory, 327–328
Utilization rate. *See* traffic intensity

V

Variables, 6
  addition and deletion of, 80–81
  deviational, 125
  excess, 17
  slack, 17
  surplus, 17
Variance, 436
Varignon frame, 239
Vector, 427–428
Vector optimization, 116–124
Vertex, 177
Vertex substitution method, 244
von Stackelberg solutions, 246–247

W

Wald's rule, 311
Weighted shortest processing time algorithm. *See WSPT* algorithm
Weighting method, 121–122
Weiszfeld method, 238–239
Workload balancing, 152–153
WSPT algorithm, 291